APPROXIMATION ALGORITHMS FOR DIVERSITY-AWARE K-NEAREST NEIGHBOR SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

As a fundamental computational primitive, Nearest Neighbor Search (NNS) retrieves the K most relevant objects for a given query, underpinning a vast array of applications like image search, retrieval-augmented generation (RAG), and recommendation systems. However, traditional NNS neglects diversity awareness, thereby suffering from severe redundancy and near-duplicate results. To tackle this limitation, diversity-aware K-nearest neighbor search (DKNNS) has been extensively studied. Despite these efforts, state-of-the-art DKNNS methods still struggle with compromised result quality, efficiency bottlenecks, or inflexible trade-offs between relevance and diversity. To address these limitations, we study a DKNNS formulation that linearly combines the relevance and diversity terms in a unified objective function, with a controllable parameter λ to adjust their relative importance. We prove that the problem is NP-hard and that, unless $\mathrm{P} = \mathrm{NP}$, no polynomial-time algorithm can approximate it with a factor of $(4/3 - \zeta)$ for any $\zeta > 0$. Then, we propose a linear scan-based algorithm, threshold-greedy scan (TGS), for DKNNS with a $(4/3 + \epsilon)$-approximation for any fixed $\epsilon > 0$. Furthermore, we integrate the threshold-greedy principle with graph-based indexes for NNS and propose multi-threshold graph search (MTGS), which achieves a $(2 + \epsilon)$-approximation under a shortcut-reachability assumption on the graph. Finally, we conduct extensive experiments on four real-world datasets. The results demonstrate that TGS achieves the best result quality for DKNNS in almost all cases. Meanwhile, MTGS runs over two orders of magnitude faster than TGS and improves upon state-of-the-art DKNNS baselines in both result quality and search efficiency.

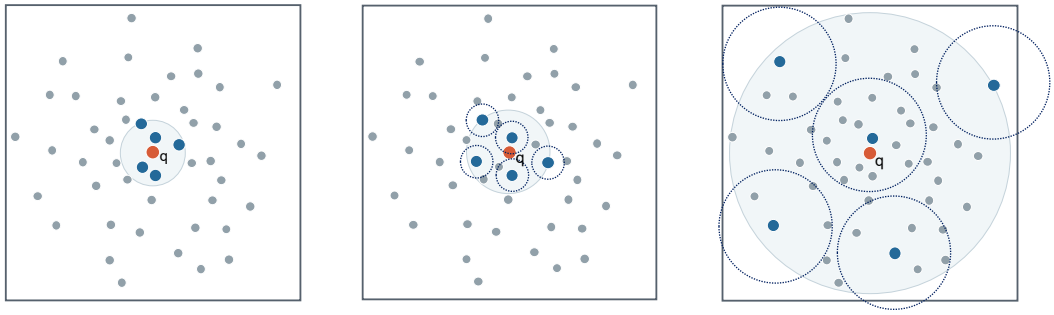
1 INTRODUCTION

Nearest neighbor search (NNS) on large-scale, high-dimensional data (Li et al., 2019) has been extensively studied in recent years for its broad applications in recommender systems (Zhao et al., 2024; Liu et al., 2024), retrieval-augmented generation (RAG) (Ma et al., 2023), image and video search (Romero et al., 2023), etc. In these applications, data objects are represented as vectors in an embedded space, and for a given query q embedded in the same space, an NNS returns the K data objects that are closest (i.e., most relevant or similar) to q.

Due to the "curse of dimensionality" (Peng et al., 2024), exact NNS becomes very inefficient in high dimensions. As such, existing studies focus primarily on approximate NNS (ANNS), which trade a slight loss in query accuracy for substantially higher efficiency. This is achieved by utilizing specific data structures, e.g., trees (Bentley, 1975; Muja & Lowe, 2009), proximity graphs (Malkov & Yashunin, 2020; Fu et al., 2019; Subramanya et al., 2019), locality-sensitive hashing (LSH) (Indyk & Motwani, 1998), and product quantization (PQ) (Jégou et al., 2011), for search space reduction. Among these, graph-based methods stand out for their high empirical performance. Nevertheless, most graph-based ANNS approaches cannot provide theoretical guarantees. Recently, a few studies (Prokhorenkova & Shekhovtsov, 2020; Indyk & Xu, 2023; Gollapudi et al., 2025; Al-Jazzazi et al., 2025) have attempted to bridge this gap by establishing worst-case bounds on the quality of NNS results for specific graph families and search algorithms.

Despite the above advances, one fundamental issue with NNS is the lack of *diversity*. Specifically, NNS results can contain highly redundant or even nearly duplicate objects, which is undesirable in

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.



(a) Ordinary NNS ($\lambda = 0$) (b) DKNNS ($\lambda = 0.3$) (c) DKNNS ($\lambda = 0.5$)

Figure 1: Illustrative examples of NNS and DKNNS (with $K = 5$) for different λ values.

real-world applications. For example, highly similar documents provide little or no new information for LLMs in RAG (Khan et al., 2026); items with only slight differences can lower user satisfaction in recommender systems (Kunaver & Požrl, 2017). This has motivated extensive studies on introducing diversity into similarity search results.

One line of studies focuses primarily on *diversity-based reranking* (Carbonell & Goldstein, 1998; Drosou & Pitoura, 2010; Abbar et al., 2013; Santos et al., 2015; Zheng et al., 2017; Matsui, 2025). A standard paradigm is to first retrieve a candidate set of size $K' \gg K$ based solely on relevance and then rerank the candidates to extract K relevant yet diverse objects. This approach suffers from an efficiency bottleneck: the candidate set must be large enough to guarantee a diverse final result, while the reranking itself also incurs substantial overhead. The other line of studies incorporates diversity into NNS itself, either as a constraint or as part of the objective. Anand et al. (2025) introduce diversity as a "color" constraint, where an object is assigned a color based on its attributes or cluster memberships, and it restricts the number of same-color objects that appear in the result. Barman et al. (2026) formulate an objective that encodes diversity through Nash and p-mean welfare. Nevertheless, both formulations rely on pre-labeled categorical attributes and do not extend to the common case of measuring relevance and diversity based on distances in a unified latent space. Zhao et al. (2026) consider NNS with a diversity constraint that requires the distance between any pair of objects in the result to be above a prespecified threshold. The threshold, however, controls the relevance-diversity trade-off indirectly and is inflexible for end users to adjust. In addition, its search method, despite using graph indexes, still incurs high computational costs.

Our Contributions To address the above limitations, we formulate the diversity-aware K-nearest neighbor search (DKNNS) problem as minimizing an objective function that linearly combines *relevance* (i.e., the average distance between the query q and all objects in the result set S) and *diversity* (the opposite of the minimum pairwise distance among the objects in S), with a controllable and tunable parameter $\lambda \in [0, 1/2]$ to balance the relative importance of both terms. Figure 1 uses examples in a two-dimensional Euclidean space to illustrate how the value of λ affects the relevance-diversity trade-off. We prove that, for any fixed $\lambda \in (0, 1/2]$, DKNNS is NP-hard even in Euclidean spaces. For general metric spaces, we show that no polynomial-time $(4/3 - \zeta)$-approximation algorithm for DKNNS exists for any fixed $\zeta \in (0, 1/3)$ unless P = NP.

Then, we propose a linear scan-based algorithm, threshold-greedy scan (TGS), for DKNNS. Generally, TGS scans all data objects in ascending order of their distances to the query q and accepts an object only if it is at a distance of at least θ from all previously selected objects. By enumerating a sequence Θ_ϵ of $O(\frac{1}{\epsilon})$ thresholds w.r.t. an error parameter $0 < \epsilon < 1$ and making a scan for each $\theta \in \Theta_\epsilon$, TGS achieves a near-optimal $(4/3 + \epsilon)$-approximation for DKNNS.

Despite its strong theoretical guarantee, TGS is computationally expensive on large-scale data. To improve its efficiency, we integrate threshold-greedy selection into graph indexes for NNS. Accordingly, we propose multi-threshold graph search (MTGS), which performs a best-first graph traversal and updates all threshold-specific candidates as each object is expanded. Our analysis establishes that, for any fixed $\epsilon > 0$, MTGS achieves a $(2+\epsilon)$-approximation for DKNNS under the assumption that the graph index is α-shortcut-reachable (Indyk & Xu, 2023).

To evaluate the empirical performance of our proposed methods, we conduct extensive experiments on four real-world datasets. The results show that TGS achieves the lowest objective values in almost all cases, but at the cost of high computational overhead due to linear scans over the whole dataset. In addition, MTGS provides a favorable quality-efficiency trade-off. Specifically, across the four datasets, it mostly achieves lower objective values than the state-of-the-art DKNNS baselines, i.e., LotusFilter (Matsui, 2025) and PSS (Zhao et al., 2026), while running up to $6.5\times$ faster than LotusFilter and $910\times$ faster than PSS.

2 Problem Formulation

Let us consider a metric space (X, d) with a distance function $d : X \times X \mapsto \mathbb{R}_{\geq 0}$. We assume no special properties of (X, d) beyond those general for any metric space (i.e., the identity of indiscernibles, non-negativity, symmetry, and the triangle inequality). Given a database $D = \{x_1, \ldots, x_n\} \subseteq X$ of n distinct objects, a query $q \in X$, and a size constraint $2 \leq K \leq n$, the (exact) K-nearest neighbor search (NNS) aims to find a size-K set of objects in D that are closest to q. Let $\mathcal{S}_K(D) := \{S \subseteq D : |S| = K\}$ denote the family of all size-K subsets of D. The NNS problem can be equivalently denoted as $\min_{S \in \mathcal{S}_K(D)} \frac{1}{K} \sum_{x \in S} d(q, x)$.

In this paper, we aim to retrieve objects that are not only close to q (i.e., *relevance*) but also dissimilar from one another (i.e., *diversity*). To measure the diversity of a set $S \subseteq D$, we follow the well-known concept of *max-min dispersion* (Carbonell & Goldstein, 1998; Drosou & Pitoura, 2010; Ravi et al., 1994), which uses the minimum distance between any pair of distinct objects within S to quantify its diversity. To formalize the trade-off between relevance and diversity, we combine them into a single objective function as

$$f_\lambda(q, S) := (1 - \lambda)\,\mathrm{dist}(q, S) - \lambda\,\mathrm{div}(S) + \Delta, \tag{1}$$

where

$$\lambda \in [0, 1/2],\ \mathrm{dist}(q, S) := \frac{1}{K} \sum_{x \in S} d(q, x),\ \mathrm{div}(S) := \min_{x \neq y \in S} d(x, y),\ \Delta := \max_{x, y \in D} d(x, y). \tag{2}$$

Since $\mathrm{div}(S) \leq \Delta$ by their definitions, we add Δ to $f_\lambda(q, S)$ to ensure its non-negativity. As Δ only depends on D and is independent of S, it does not affect the relative ranking of any pair of sets w.r.t. $f_\lambda(q, \cdot)$. Introducing Δ allows for a multiplicative approximation based on $f_\lambda(q, S)$, as will be defined below, without regularization. In addition, the parameter λ determines the relative importance of relevance and diversity. When $\lambda = 0$, $f_\lambda(q, S)$ reduces to the objective of an ordinary NNS. As λ grows, the objective favors more diverse results. We limit λ to the range $[0, 1/2]$ so that the relevance term remains at least as important as the diversity term, since our primary goal is to introduce diversity into NNS, rather than the other way around. Given the above concepts, we now formally define the diversity-aware K-nearest neighbor search (DKNNS) problem.

Definition 1 (DKNNS)**.** For a database $D \subseteq X$, a query $q \in X$, a size constraint $2 \leq K \leq n$, and a parameter $\lambda \in [0, 1/2]$, DKNNS returns a size-K subset of D that minimizes $f_\lambda(q, S)$ in Eq. 1, i.e., $S^* = \arg\min_{S \in \mathcal{S}_K(D)} f_\lambda(q, S)$.

In a metric space, however, the relevance and diversity terms are constrained by the triangle inequality. The following lemma captures this relationship and will be used in subsequent analysis.

Lemma 2.1. *For every set $S \in \mathcal{S}_K(D)$, it holds that* $\mathrm{dist}(q, S) \geq \mathrm{div}(S)/2$.

See Appendix C.1 for the proof.

Next, we analyze the computational hardness of DKNNS. The following theorems indicate the NP-hardness of DKNNS and its hardness of approximation.

Theorem 2.2. *For any $\lambda \in (0, 1/2]$, DKNNS in the Euclidean space $\mathbb{R}^{n^2}$ is NP-hard.*

Theorem 2.3. *Assuming* P $\neq$ NP*, for every fixed $\zeta \in (0, 1/3)$, no polynomial-time $(4/3 - \zeta)$-approximation algorithm exists for DKNNS.*

We prove Theorems 2.2 and 2.3 by reducing two well-known NP-hard problems, namely Independent Set and Clique (Karp, 1972), to DKNNS (see Appendices C.2 & C.3 for the reduction procedures). Given the above results, we focus on approximation algorithms for DKNNS. Specifically, for any $c \geq 1$, a solution $\widehat{S} \in \mathcal{S}_K(D)$ is a c-approximation if $f_\lambda(q, \widehat{S}) \leq c \cdot f_\lambda(q, S^*)$. An algorithm is c-approximate for DKNNS if it provides a c-approximation for any DKNNS instance.

Algorithm 1 Threshold-Greedy Scan (TGS)

Input: Database D, query q, size constraint K, parameters λ, ϵ
Output: A set $S_{\mathrm{TGS}} \subseteq D$ such that $|S_{\mathrm{TGS}}| = K$

```
Sort D by non-decreasing order of d(q, x) as D'
if δ_min = δ_max then return the first K objects in D' as S_TGS
Construct the sequence of thresholds Θ_ε as Eq. 3
for j = 0, ..., J do
    T_{θ_j} ← ∅
    for all x ∈ D' do
        if d(x, y) ≥ θ_j for every y ∈ T_{θ_j} then
            T_{θ_j} ← T_{θ_j} ∪ {x}
            if |T_{θ_j}| = K then break
    if |T_{θ_j}| < K then break
return S_TGS = arg min_{θ_j ∈ Θ_ε : |T_{θ_j}| = K} f_λ(q, T_{θ_j})
```

3 Linear Scan-based Algorithm for DKNNS

Threshold-Greedy Scan We propose Threshold-Greedy Scan (TGS), a $(4/3+\epsilon)$-approximation DKNNS algorithm for any $\epsilon \in (0,1)$. Note that such an approximation is nearly the lower bound in Theorem 2.3. Generally, TGS performs a linear scan to sort all objects in D in ascending order of their distance to the query q and considers each object one by one in that order. An object x is accepted into the solution if it is at a distance of at least θ from each object already selected, for a fixed threshold $\theta > 0$. In this way, TGS favors the objects closer to q while enforcing diversity.

In the TGS procedure, the choice of θ is critical to achieving a good solution. Ideally, θ would be set on the scale of the optimal diversity $\mathrm{div}^* = \mathrm{div}(S^*)$, which is unknown in advance. TGS thus enumerates a sequence of thresholds over $[\delta_{min}/2, \delta_{max}/2]$, where $\delta_{min} = \min_{x \neq y \in D} d(x,y)$ and $\delta_{max} = \Delta = \max_{x \neq y \in D} d(x,y)$, so that at least one value in the sequence can approximate $\mathrm{div}^*/2$ within a bounded error. For an error parameter $0 < \epsilon < 1$, let $J = \lceil 1/\epsilon \rceil$ and $h_\epsilon = (\delta_{\max} - \delta_{\min})/(2J)$. We can generate a sequence of thresholds $\Theta_\epsilon = \{\theta_0, \ldots, \theta_J\}$ as:

$$\theta_j = \delta_{\min}/2 + j h_\epsilon, \ \forall j = 0, \ldots, J. \tag{3}$$

Obviously, $h_\epsilon \leq \epsilon \delta_{\max}/2$, and $|\Theta_\epsilon| = J + 1 = O(1/\epsilon)$.

The full procedure of TGS is summarized in Algorithm 1. It first sorts all objects in D in non-decreasing order of relevance as D'. Then, it generates Θ_ϵ and considers each threshold within it in ascending order. For each $\theta_j \in \Theta_\epsilon$, it scans each object of D' in order and accepts any object that is at least θ_j distance away from every object already selected into the candidate T_{θ_j} w.r.t. θ_j. Once T_{θ_j} contains K objects, the scan w.r.t. θ_j terminates immediately. In this way, it satisfies that $\mathrm{div}(T_{\theta_j}) \geq \theta_j$ by construction. When a scan exhausts D' without selecting K objects, TGS no longer considers larger thresholds. Finally, TGS compares all candidates of size K and picks the one with the smallest objective value as the final solution.

Theoretical Analysis We first provide a general property for any threshold $\theta \leq \mathrm{div}^*/2$. Given any $T \subseteq D$ and $\theta > 0$, we use $F(T, \theta)$ to denote the set of objects in $D \setminus T$ that are at a distance of at least θ from every object in T. We can establish the following two results. First, when $\theta \leq \mathrm{div}^*/2$, a feasible object always exists at each step, and thus a candidate of size K can be found. Second, if each step selects a β-approximate nearest object, i.e., its distance to q is at most β times the minimum over the current feasible set, the resulting set has an average distance of at most β times that of the optimal solution. Formally, we have the following lemma:

Lemma 3.1. *Fix $0 < \theta \leq \mathrm{div}^*/2$ and $\beta \geq 1$. Let $T_{i-1} = \{p_1, \ldots, p_{i-1}\}$ and $F_i = F(T_{i-1}, \theta)$. Suppose that at the i-th step, the selected object $p_i \in F_i$ satisfies $d(q, p_i) \leq \beta \min_{z \in F_i} d(q, z)$. Then, we have $F_i \neq \emptyset$ for each $1 \leq i \leq K$, and $T = \{p_1, \ldots, p_K\}$ satisfies that $\mathrm{dist}(q, T) \leq \beta\, \mathrm{dist}(q, S^*)$ and $\mathrm{div}(T) \geq \theta$.*

Next, we apply Lemma 3.1 to TGS. For a given threshold, TGS accepts the first feasible object in each iteration. Thus, we have $\beta = 1$ in Lemma 3.1 for TGS. Then, we choose the largest $\theta \in \Theta_\epsilon$ that does not exceed $\mathrm{div}^*/2$ and have the following lemma:

Lemma 3.2. *Let $\widehat{\theta}$ be the largest threshold $\theta \in \Theta_\epsilon$ such that $\theta \leq \mathrm{div}^*/2$. Then, we have*

$$\max\left\{\frac{\delta_{min}}{2}, \frac{\mathrm{div}^*}{2} - h_\epsilon\right\} \leq \widehat{\theta} \leq \frac{\mathrm{div}^*}{2}. \tag{4}$$

Given the above lemmas, the candidates computed by TGS must contain $T_{\widehat{\theta}}$. By comparing S_{TGS} with $T_{\widehat{\theta}}$, we can obtain the approximation result in the following theorem.

Theorem 3.3. *For any fixed $0 < \epsilon < 1$, TGS is $(4/3 + \epsilon)$-approximate for DKNNS.*

Finally, the time complexity of TGS is $O(n(\log n + K/\epsilon))$. The proofs and analyses of the above results are provided in Appendices C.4–C.7.

4 Graph-Based Algorithm for DKNNS

TGS requires scanning the full database to provide an approximate solution. This becomes computationally expensive on large-scale data. In this section, we consider how to use graph indexes for approximate NNS (Fu et al., 2019; Malkov & Yashunin, 2020; Subramanya et al., 2019) to avoid repeatedly scanning the database while still retaining an approximation bound for DKNNS.

Graph Index for NNS Graph indexes have demonstrated high efficiency and scalability for NNS. Specifically, a graph index represents the database D as a proximity graph $\mathcal{G} = (D, E)$, where each vertex is an object in D and edges connect close objects to form a navigable structure. Graph indexes rely on greedy and beam search to perform NNS. Starting from an entry vertex, a greedy search evaluates the distances of the current vertex's neighbors and moves to the one closest to the query q. In this way, it can reach q's neighborhood quickly after traversing only a small fraction of D. A beam search is built on and extends the greedy search. It maintains a queue of objects ordered by their distances to q, repeatedly expands the most promising unexpanded vertex, and retains at most a fixed number of candidates (called the beam width) in the queue. However, both search methods do not take into account diversity and cannot be directly used for DKNNS.

Furthermore, we ground our analysis in the certified α-shortcut-reachability property of $\mathcal{G}$ proposed by Indyk & Xu (2023). Specifically, let $N^+(x)$ denote the out-neighborhood of $x \in D$ on $\mathcal{G}$. $\mathcal{G}$ is assumed to be supplied with a constant factor $\alpha > 1$ such that for every pair of vertices $x, z \in D$, either $(x, z) \in E$ or there exists $y \in N^+(x)$ with

$$\alpha \cdot d(y, z) \leq d(x, z). \tag{5}$$

Such a vertex y is called a "shortcut" from x to z. Since $\alpha > 1$, a greedy search from any source vertex can reach any target vertex through a sequence of steps that progressively reduce the distance to it. This implies that $\mathcal{G}$ is strongly connected. Note that such a property requires a specific graph structure, e.g., unlimited-degree, slow-preprocessing DiskANN graphs (Subramanya et al., 2019), but does not extend to bounded-degree graphs, such as HNSW (Malkov & Yashunin, 2020).

Multi-Threshold Graph Search We introduce Multi-Threshold Graph Search (MTGS) to incorporate the threshold-greedy principle of TGS into graph-indexed retrieval. Conventional beam search maintains a bounded queue and ranks vertices only by their distances to the query, without explicitly considering the pairwise diversity of the returned set. Running an independent graph search for every threshold would repeatedly visit the same vertices and lead to duplicate computations. MTGS instead evaluates all thresholds through a single shared graph traversal.

Given the graph $\mathcal{G} = (D, E)$, MTGS first constructs the threshold sequence Θ_ϵ according to Eq. 3 and initializes an empty candidate T_θ for every $\theta \in \Theta_\epsilon$. It then starts from the entry vertex s, evaluates $d(q, s)$, records $D_{\mathrm{seen}} = \{s\}$ and $D_{\mathrm{exp}} = \emptyset$, and inserts s into a min-heap Q keyed by the distance to q.

While Q is nonempty, MTGS extracts the discovered but unexpanded vertex x with the smallest distance to q and adds it to D_{exp}. The extracted vertex is then passed to the candidate for each threshold that has not yet collected K objects. For each $\theta \in \Theta_\epsilon$, MTGS accepts x into T_θ if $d(x, z) \geq \theta$ for every $z \in T_\theta$. Thus, every completed candidate satisfies $\mathrm{div}(T_\theta) \geq \theta$ by construction. Once $|T_\theta| = K$, the candidate is complete and no longer needs to process subsequent vertices.

Algorithm 2 Multi-Threshold Graph Search (MTGS)

Input: Graph $\mathcal{G} = (D, E)$, entry vertex s, query q, cardinality K, and parameters λ, ϵ
Output: A set $S_{\mathrm{MTGS}} \subseteq D$ such that $|S_{\mathrm{MTGS}}| = K$

```
Construct Θ_ε according to Eq. 3
for all θ ∈ Θ_ε do
    Initialize T_θ ← ∅
Initialize D_seen ← {s}, D_exp ← ∅ and a min-heap Q ← {s}
while Q ≠ ∅ do
    x ← ExtractMin(Q)
    D_exp ← D_exp ∪ {x}
    for all θ ∈ Θ_ε do
        if |T_θ| < K and d(x, z) ≥ θ for every z ∈ T_θ then
            T_θ ← T_θ ∪ {x}
    for all y ∈ N⁺(x) do
        if y ∉ D_seen then
            D_seen ← D_seen ∪ {y} and Insert(Q, y)
C ← {T_θ : θ ∈ Θ_ε ∧ |T_θ| = K}
return S_MTGS = arg min_{T∈C} f_λ(q, T)
```

After updating the candidates, MTGS expands x by scanning its outgoing neighbors. For every $y \in N^+(x)$ that is not already in D_{seen}, it evaluates and caches $d(q, y)$, adds y to D_{seen}, and inserts y into Q. The membership test ensures that each vertex is inserted into the heap at most once, while adding x to D_{exp} ensures that its outgoing edges are scanned only once. Therefore, all threshold candidates share the same vertex discoveries, query-distance evaluations, and graph expansions; only their pairwise feasibility checks are maintained separately.

The traversal continues until Q becomes empty. MTGS then collects all complete candidates as $\mathcal{C} = \{T_\theta : \theta \in \Theta_\epsilon \text{ and } |T_\theta| = K\}$ and returns $S_{\mathrm{MTGS}} = \arg\min_{T \in \mathcal{C}} f_\lambda(q, T)$. Under the certified α-shortcut-reachability assumption, the uncapped traversal eventually visits the entire database D. This is because, for any distinct $x, z \in D$, either x has a direct edge to z, or it has an outgoing neighbor y satisfying $d(y, z) \leq d(x, z)/\alpha < d(x, z)$. Repeating such shortcut steps must eventually reach z because D is finite. Hence, $\mathcal{G}$ is strongly connected. Since MTGS never discards a discovered but unexpanded vertex, $Q = \emptyset$ implies $D_{\mathrm{seen}} = D_{\mathrm{exp}} = D$.

Theoretical Analysis Next, we establish MTGS's approximation guarantee under the α-shortcut-reachability assumption and analyze its time complexity. Specifically, we prove that MTGS achieves a $(2 + \epsilon)$-approximation for any fixed $\epsilon > 0$ and obtain a complexity bound that separates the cost of the shared graph traversal from that of the threshold-specific feasibility checks.

Theorem 4.1. *For a graph index that satisfies the certified α-shortcut-reachability, any fixed $\lambda \in [0, 1/2]$, $0 < \epsilon < 1$, MTGS is a $(2 + \epsilon)$-approximation for DKNNS.*

Finally, the time complexity of MTGS is $O(E_{\mathrm{exp}} + M \log M + MK\epsilon^{-1})$, where E_{exp} is the total number of outgoing edges scanned during graph expansion and $M = |D_{\mathrm{seen}}|$ is the number of objects discovered during the search procedure. The proofs and analyses of the above results are provided in Appendices C.8 – C.9.

5 Experiments

5.1 Experimental Setup

Environment All experiments were conducted on a server with two Intel(R) Xeon(R) Silver 4314 processors (32 physical cores and 64 hardware threads in total) and 256 GiB of DDR4 memory running Ubuntu 20.04.3 LTS. We compiled the C/C++ code with GCC/G++ 13.1.0 and used Python 3.11. We did not use any GPU in the experiments.

Table 1: Statistics of datasets used in the experiments.

Dataset	Dimension	#Objects	#Queries	Distance	Modality	Distribution
GIST	960	1,000,000	100	Euclidean	Image	Skewed
MSong	420	983,185	100	Euclidean	Audio	Skewed
GloVe	200	1,183,514	100	Euclidean	Text	Normal
SIFT	128	1,000,000	100	Angular	Image	Skewed

Datasets We evaluate all methods on four standard benchmark datasets for NNS, as summarized in Table 1. GIST, MSong, and GloVe are evaluated using the Euclidean distance. SIFT is evaluated using the angular distance after performing an ℓ_2-normalization for all vectors. For normalized vectors, we use $d_{\mathrm{ang}}(x, y) = \arccos(\mathrm{clip}(x^\top y, -1, 1))/\pi$. For each dataset, we use the same distance to build the graph and apply it consistently across all experiments.

Compared Methods We compare our proposed methods, TGS and MTGS, against several state-of-the-art DKNNS methods: (i) LotusFilter (Matsui, 2025), which uses a learned cutoff table to accelerate two-stage filtering for DKNNS; (ii) PSS (Zhao et al., 2026), which formulates DKNNS as an NNS subject to the constraint that any pair of objects in the result must be at a distance of ϵ_{pss}; and (iii) Beam Search (BS) (Malkov & Yashunin, 2020), a classic non-diversity-aware graph-based nearest neighbor search method. We do not include the methods in (Barman et al., 2026) and (Anand et al., 2025) since their problem definitions do not align with ours.

All graph-based methods share the same HNSW graph (Malkov & Yashunin, 2020). The HNSW graph construction sets the maximum number of neighbors to $R = 64$ for every graph-based method, including the first-stage candidate generation of LotusFilter. TGS is the only full-scan method and does not use the graph. Note that the theoretical analysis assumes the α-shortcut-reachability property of the graph, yet HNSW used in the experiments does not necessarily meet such an assumption. We additionally test the compatibility of different graphs for NNS with our MTGS method.

Configuration The default setting uses $K = 50$, $\lambda \in \{0.2, 0.5\}$, and $\epsilon = 0.005$. LotusFilter uses a candidate budget $C = 5,000$. For MTGS, to bound the online search cost, the level-0 traversal is truncated after at most $B = \max\{K, L\}$ vertex expansions, and we set $L = 5K$. For a fair and reproducible comparison, we determine the diversity threshold for each baseline offline. For PSS, we select the threshold ϵ_{PSS} such that the estimated average degree of the induced conflict graph satisfies $\phi(\epsilon_{\mathrm{PSS}}) \approx 100$, as suggested by the original paper (Zhao et al., 2026). For LotusFilter, we follow the preprocessing procedure of (Matsui, 2025) and select ϵ_{LF} through an iterative coarse-to-fine grid search on training pseudo-queries, separately for each dataset and (K, S, λ) configuration.

Evaluation Metrics We report the objective value (Obj.), average distance to query q (Dist.), minimum pairwise distance (Div.), and query latency (Time). We average all metrics over 100 queries. Since the dataset diameter Δ is an additive constant identical for all methods that does not affect comparisons, we report the objective values that omit the Δ term throughout all experiments. Graph construction and one-time preprocessing are excluded from the query time.

5.2 Experimental Results

Overall Performance We first conduct experimental evaluations with $K = 50$ and $\lambda = 0.2, 0.5$ under the default configurations. The results of LotusFilter are omitted on SIFT because it is specific to the Euclidean distance. The results are reported in Table 2.

Generally, the results reveal a clear trade-off between quality and latency. TGS attains the lowest objective values in all eight dataset-λ settings but requires 0.83–8.04 seconds per query. Beam Search (BS) is the fastest method, but it comes with consistently lower diversity and a higher objective value than MTGS. In contrast, MTGS completes queries in only 13.05–49.44 ms while retaining substantially better solution quality. Compared with TGS, MTGS is approximately 38–407 times faster at $\lambda = 0.2$ and 38–269 times faster at $\lambda = 0.5$. At $\lambda = 0.2$, its objective value is within 0.3–0.8% of TGS on GIST, MSong, and SIFT, and within 7.3% on GloVe. Under the higher diversity weight of

Table 2: Performance of each method across the four datasets with $K = 50$ and $\lambda = 0.2, 0.5$ under default configurations. Best results are in **bold**, and second-best results are underlined.

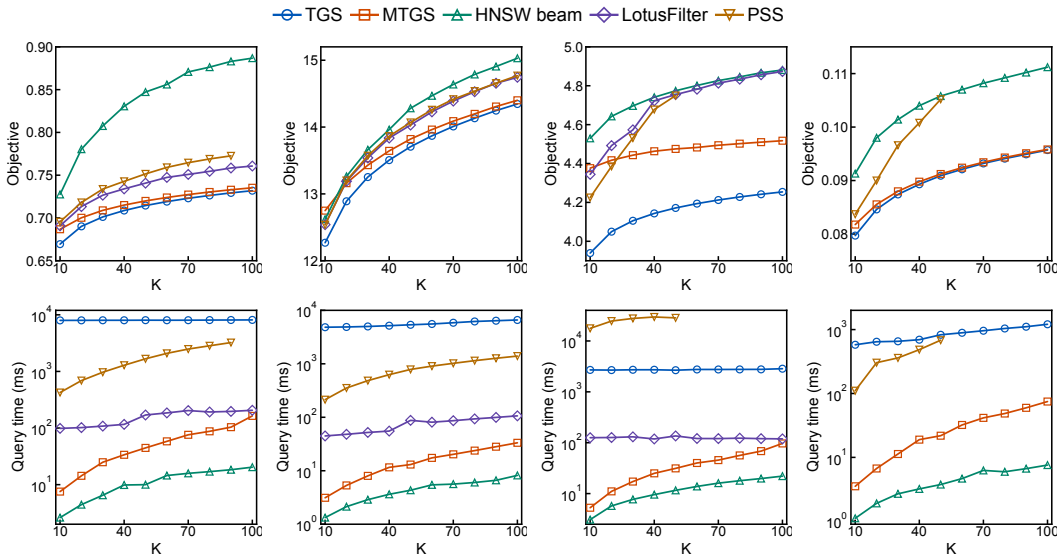
Dataset	Method	$\lambda = 0.2$				$\lambda = 0.5$			
		Obj.	Time (ms)	Div.	Dist.	Obj.	Time (ms)	Div.	Dist.
GIST	TGS	**0.7145**	8041.8	1.0210	1.1483	**-0.0533**	8023.5	**2.2217**	2.1151
	MTGS	0.7199	44.56	**1.0212**	1.1552	0.0572	49.44	1.0508	1.1652
	BS	0.8472	**9.993**	0.2762	**1.1280**	0.4259	**31.33**	0.2762	**1.1280**
	LotusFilter	0.7407	167.93	0.8303	1.1334	0.1548	126.59	0.8318	1.1414
	PSS	0.7515	1671.7	0.7696	1.1318	0.1811	1664.7	0.7696	1.1318
MSong	TGS	**13.7062**	5306.8	**16.3370**	21.2171	**-1.0247**	5326.8	**76.5588**	74.5093
	MTGS	13.8209	13.05	15.4670	21.1429	2.5329	19.78	16.3633	21.4291
	BS	14.2812	**4.3198**	11.8364	**20.8106**	4.4871	**4.1659**	11.8364	**20.8106**
	LotusFilter	14.0303	85.04	13.4606	20.9031	3.2920	63.66	14.7321	21.3160
	PSS	14.0689	785.01	13.8783	21.0557	3.5887	777.32	13.8784	21.0557
GloVe	TGS	**4.1715**	2663.1	**4.9486**	6.4515	**0.0003**	2711.0	**9.7239**	9.7292
	MTGS	4.4751	31.30	2.5907	6.2416	1.8212	34.72	2.6039	6.2463
	BS	4.7741	**11.59**	0.7356	6.1515	2.7080	**23.2**	0.7356	**6.1515**
	LotusFilter	4.7547	135.9	0.7997	**6.1433**	2.1242	140.7	1.9233	6.1717
	PSS	4.7495	28499.4	4.2256	6.9933	1.3838	28483.1	4.2256	6.9933
SIFT	TGS	**0.0909**	828.99	0.1244	0.1447	**-0.0050**	829.96	**0.2279**	0.2178
	MTGS	0.0912	21.84	**0.1253**	0.1453	0.0054	22.08	0.1405	0.1514
	BS	0.1058	**3.758**	0.0321	**0.1402**	0.0540	**13.401**	0.0321	**0.1402**
	LotusFilter	–	–	–	–	–	–	–	–
	PSS	0.1052	682.32	0.1191	0.1612	0.0210	686.50	0.1191	0.1612

$\lambda = 0.5$, TGS yields a larger objective-value advantage, particularly on MSong and GloVe; however, MTGS still achieves the second-lowest objective on three of the four datasets. Relative to BS, MTGS incurs a latency overhead of 1.5–5.8 times but consistently achieves both higher diversity and a lower objective value while keeping query time within tens of milliseconds.

Across all six settings where LotusFilter is available, MTGS achieves a lower objective value in every case while being 2.6–6.5 times faster. Compared with PSS, MTGS obtains lower objective values in seven of the eight settings and is approximately 31–911 times faster. The only exception is GloVe at $\lambda = 0.5$, where the improvement provided by PSS comes with an approximately 820-fold latency increase: PSS requires 28.48 seconds per query, compared with only 34.72 ms for MTGS. Therefore, MTGS provides a significantly better overall quality-latency trade-off: it largely preserves the solution quality of TGS at $\lambda = 0.2$, dominates LotusFilter in every available comparison, and outperforms PSS in both objective value and query latency in seven of the eight settings.

Effect of Size Constraint K To examine how the output size K affects solution quality and query efficiency, we vary $K \in \{10, 20, \ldots, 100\}$ while fixing $\lambda = 0.2$ and keeping all other configurations unchanged. The objective value and query-time results are shown in Figure 2. For PSS, it fails to return exactly K objects in some cases and, thus, is omitted for the corresponding K values.

The query time exhibits a stable computational hierarchy as K increases. Beam search is consistently the fastest method, followed by MTGS. Across the four datasets, the latency of MTGS grows from only a few milliseconds at $K = 10$ to tens or, at most, low hundreds of milliseconds at $K = 100$. In comparison, TGS requires hundreds of milliseconds to several seconds due to its exhaustive access, while LotusFilter and PSS remain substantially slower than MTGS over the reported ranges. PSS is particularly expensive on GloVe, and its latency increases markedly with K on the other datasets. These results therefore show that MTGS maintains a clear efficiency advantage over the diversity-aware baselines as the result size grows. The objective value generally increases with K for all methods, reflecting the greater difficulty of jointly maintaining relevance and diversity in a larger result set. TGS consistently achieves the lowest objective, whereas MTGS follows a smoother and more gradual increase than most competing approximate methods. Consequently,



(a) GIST (b) MSong (c) GloVe (d) SIFT

Figure 2: Objective values and query time of each method with varying K.

Table 3: Effect of the graph index choice on the performance of MTGS.

Method	Obj.	Time (ms)	Div.	Dist.
HNSW (Malkov & Yashunin, 2020)	0.7199	44.5626	1.0212	1.1552
NSG (Fu et al., 2019)	0.7260	40.6431	1.0284	1.1646
DiskANN (Subramanya et al., 2019)	0.7229	71.5679	1.0273	1.1604

the objective gap between MTGS and TGS narrows as K grows. Their curves become particularly close on GIST and SIFT, and MTGS also approaches TGS on MSong at larger K. Although the gap remains more obvious on GloVe, MTGS still achieves a lower objective than the other methods over most of the evaluated range. Taken together, these results show that MTGS scales favorably with the output size: its solution quality increasingly approaches that of TGS, while its query time remains much closer to that of beam search.

Effect of Graph Index To assess MTGS's compatibility with different proximity graphs, we evaluate HNSW, NSG, and DiskANN on GIST while keeping all other settings fixed. The results are reported in Table 3. The three graph indexes produce very close solution quality, with all measures differing by less than 1%. HNSW achieves the lowest objective value and average distance to query, whereas NSG provides the highest diversity and reduces query time by 8.8% relative to HNSW. DiskANN achieves intermediate values for all three quality metrics but requires 71.57 ms per query, about 1.6 times the latency of HNSW and 1.8 times that of NSG. These results indicate that MTGS keeps robust solution quality w.r.t. the evaluated graph index, yet its query efficiency remains affected by the graph structure.

6 Conclusion

This paper studied diversity-aware K-nearest-neighbor search (DKNNS). We proved that the problem is NP-hard and that no polynomial-time $(4/3 - \zeta)$-approximation exists unless P $=$ NP. We then proposed threshold-greedy scan (TGS) to obtain a $(4/3 + \epsilon)$-approximation. We further proposed multi-threshold graph search (MTGS) to integrate the threshold-greedy principle with graph-indexed retrieval. Under the α-shortcut-reachability assumption, MTGS achieved a $(2 + \epsilon)$-approximation. Finally, extensive experiments on four real-world datasets validated the effectiveness and efficiency of our algorithms.

AI USE STATEMENT

In this work, we used ChatGPT 5.6 to generate the synthetic datasets used in the supplementary experiments, assist with implementing experimental code, and edit author-written manuscript text to improve readability, grammar, and English style. All research ideas, problem formulations, algorithm designs, mathematical claims, theoretical analyses, and proofs were developed and written by the authors without generative AI assistance. All AI-generated or AI-assisted datasets, code, and textual edits were manually reviewed and validated by the authors, and all AI-assisted code was tested before use. The authors take full responsibility for the accuracy, integrity, and final content of this work.

REPRODUCIBILITY STATEMENT

The code is available in an anonymized repository: `https://anonymous.4open.science/r/Approximation-Algorithms-for-Diversity-Aware-K-Nearest-Neighbor-Search-34AD/`. We document the problem formulation, algorithms, datasets, experimental settings, evaluation metrics, implementation environment, and parameter choices. We also clearly state the assumptions underlying our theoretical results and provide complete proofs of all theoretical claims, complexity analyses, and additional experimental results.

REFERENCES

Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, Sepideh Mahabadi, and Kasturi Varadarajan. Diverse near neighbor problem. In *Proceedings of the 29th Annual Symposium on Computational Geometry*, pp. 207–214, 2013. doi: 10.1145/2462356.2462401.

Zeinab Abbassi, Vahab S. Mirrokni, and Mayur Thakur. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 32–40, 2013. doi: 10.1145/2487575.2487636.

Sepideh Aghamolaei, Majid Farhadi, and Hamid Zarrabi-Zadeh. Diversity maximization via composable coresets. In *Proceedings of the 27th Canadian Conference on Computational Geometry*, pp. 38–48, 2015. URL `http://www.cccg.ca/proceedings/2015/47.pdf`.

Yousef Al-Jazzazi, Haya Diwan, Jinrui Gou, Cameron Musco, Christopher Musco, and Torsten Suel. Distance adaptive beam search for provably accurate graph-based nearest neighbor search. In *Advances in Neural Information Processing Systems*, volume 38, pp. 110958–110987, 2025. doi: 10.52202/085713-3346.

Daichi Amagata. Diversity maximization in the presence of outliers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10):12338–12345, 2023. doi: 10.1609/aaai.v37i10.26454.

Piyush Anand, Piotr Indyk, Ravishankar Krishnaswamy, Sepideh Mahabadi, Vikas C. Raykar, Kirankumar Shiragur, and Haike Xu. Graph-based algorithms for diverse similarity search. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 1483–1504, 2025. URL `https://proceedings.mlr.press/v267/anand25a.html`.

Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. Quicker ADC: Unlocking the hidden potential of product quantization with SIMD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1666–1677, 2021. doi: 10.1109/TPAMI.2019.2952606.

Dmitry Baranchuk, Artem Babenko, and Yury Malkov. Revisiting the inverted indices for billion-scale approximate nearest neighbors. In *Computer Vision – ECCV 2018, 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XII*, pp. 202–216, 2018. doi: 10.1007/978-3-030-01258-8_13.

Siddharth Barman, Nirjhar Das, Shivam Gupta, and Kirankumar Shiragur. Welfarist formulations for diverse similarity search. In *International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=UWhOUrsgkA`.

Christian Bauckhage, Rafet Sifa, and Stefan Wrobel. Adiabatic quantum computing for max-sum diversification. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 343–351, 2020. doi: 10.1137/1.9781611976236.39.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. doi: 10.1145/361002.361007.

Michele Borassi, Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghadam. Better sliding window algorithms to maximize subadditive and diversity objectives. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 254–268, 2019. doi: 10.1145/3294052.3319701.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336, 1998. doi: 10.1145/290941.291025.

Matteo Ceccarello, Andrea Pietracaprina, Geppino Pucci, and Eli Upfal. MapReduce and streaming algorithms for diversity maximization in metric spaces of bounded doubling dimension. *Proceedings of the VLDB Endowment*, 10(5):469–480, 2017. doi: 10.14778/3055540.3055541.

Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. Fast coreset-based diversity maximization under matroid constraints. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 81–89, 2018. doi: 10.1145/3159652.3159719.

Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Max-sum diversity via convex programming. In *Proceedings of the 32nd International Symposium on Computational Geometry*, pp. 26:1–26:14, 2016. doi: 10.4230/LIPIcs.SoCG.2016.26.

Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Local search for max-sum diversification. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 130–142, 2017. doi: 10.1137/1.9781611974782.9.

Barun Chandra and Magnús M. Halldórsson. Approximation algorithms for dispersion problems. *Journal of Algorithms*, 38(2):438–465, 2001. doi: 10.1006/jagm.2000.1145.

Laming Chen, Guoxin Zhang, and Hanning Zhou. Fast greedy MAP inference for determinantal point process to improve recommendation diversity. In *Advances in Neural Information Processing Systems*, volume 31, pp. 5627–5638, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/dbbf603ff0e99629dda5d75b6f75f966-Paper.pdf.

Haya Diwan, Jinrui Gou, Cameron Musco, Christopher Musco, and Torsten Suel. Navigable graphs for high-dimensional nearest neighbor search: Constructions and limits. In *Advances in Neural Information Processing Systems*, volume 37, pp. 59513–59531, 2024. doi: 10.52202/079017-1901.

Marina Drosou and Evaggelia Pitoura. Search result diversification. *ACM SIGMOD Record*, 39(1): 41–47, 2010. doi: 10.1145/1860702.1860709.

Marina Drosou and Evaggelia Pitoura. Diverse set selection over dynamic data. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1102–1116, 2014. doi: 10.1109/TKDE.2013.44.

Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment*, 12(5):461–474, 2019. doi: 10.14778/3303753.3303754.

Siddharth Gollapudi, Ravishankar Krishnaswamy, Kirankumar Shiragur, and Harsh Wardhan. Sort before you prune: Improved worst-case guarantees of the DiskANN family of graphs. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 19787–19808, 2025. URL https://proceedings.mlr.press/v267/gollapudi25a.html.

Refael Hassin, Shlomi Rubinstein, and Arie Tamir. Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21(3):133–137, 1997. doi: 10.1016/S0167-6377(97)00034-5.

Kohei Hirata, Daichi Amagata, Sumio Fujita, and Takahiro Hara. Solving diversity-aware maximum inner product search efficiently and effectively. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 198–207, 2022. doi: 10.1145/3523227.3546779.

Qiang Huang, Yanhao Wang, Yiqun Sun, Anthony K. H. Tung, and Jun Yu. Balancing relevance and diversity in k-maximum inner product search. *The VLDB Journal*, 35(4):32, 2026. doi: 10.1007/S00778-026-00982-8.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998. doi: 10.1145/276698.276876.

Piotr Indyk and Haike Xu. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. In *Advances in Neural Information Processing Systems*, volume 36, pp. 66239–66256, 2023. doi: 10.52202/075280-2891.

Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 100–108, 2014. doi: 10.1145/2594538.2594560.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger (eds.), *Complexity of Computer Computations*, pp. 85–103. Springer, Boston, MA, 1972. doi: 10.1007/978-1-4684-2001-2_9.

Saadat Hasan Khan, Spencer Hong, Jingyu Wu, Kevin Lybarger, Youbing Yin, Erin Babinsky, and Daben Liu. DF-RAG: Query-aware diversity for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2873–2894, 2026. doi: 10.18653/v1/2026.findings-eacl.150.

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123:154–162, 2017.

Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488, 2019. doi: 10.1109/TKDE.2019.2909204.

Qidong Liu, Jiaxi Hu, Yutian Xiao, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Qing Li, and Jiliang Tang. Multimodal recommender systems: A survey. *ACM Computing Surveys*, 57(2):26:1–26:17, 2024. doi: 10.1145/3695461.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315, 2023. doi: 10.18653/v1/2023.emnlp-main.322.

Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014. doi: 10.1016/j.is.2013.10.006.

Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473.

Yusuke Matsui. LotusFilter: Fast diverse nearest neighbor search via a learned cutoff table. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30430–30439, 2025. doi: 10.1109/CVPR52734.2025.02833.

Zafeiria Moumoulidou, Andrew McGregor, and Alexandra Meliou. Diverse data selection under fairness constraints. In *24th International Conference on Database Theory (ICDT 2021)*, pp. 13:1–13:25, 2021. doi: 10.4230/LIPIcs.ICDT.2021.13.

Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications (VISAPP)*, pp. 331–340, 2009. doi: 10.5220/0001787803310340.

Yutaro Oguri and Yusuke Matsui. General and practical tuning method for off-the-shelf graph-based index: SISAP indexing challenge report by team UTokyo. In *Similarity Search and Applications — 16th International Conference, SISAP 2023, A Coruña, Spain, October 9–11, 2023, Proceedings*, pp. 273–281, 2023. doi: 10.1007/978-3-031-46994-7_23.

Dehua Peng, Zhipeng Gui, and Huayi Wu. Interpreting the curse of dimensionality from distance concentration and manifold effect. *arXiv:2401.00422*, 2024. URL https://arxiv.org/abs/2401.00422.

Liudmila Prokhorenkova and Aleksandr Shekhovtsov. Graph-based nearest neighbor search: From practice to theory. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7803–7813, 2020. URL https://proceedings.mlr.press/v119/prokhorenkova20a.html.

Vidyadhar Rao, Prateek Jain, and C. V. Jawahar. Diverse yet efficient retrieval using locality sensitive hashing. In *Proceedings of the 2016 ACM International Conference on Multimedia Retrieval*, pp. 189–196, 2016. doi: 10.1145/2911996.2911998.

Sekharipuram S. Ravi, Daniel J. Rosenkrantz, and Giri Kumar Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994. doi: 10.1287/opre.42.2.299.

Francisco Romero, Caleb Winston, Johann Hauswald, Matei Zaharia, and Christos Kozyrakis. Zelda: Video analytics using vision-language models. *arXiv:2305.03785*, 2023. URL https://arxiv.org/abs/2305.03785.

Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. Search result diversification. *Foundations and Trends in Information Retrieval*, 9(1):1–90, 2015. doi: 10.1561/1500000040.

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 *Advances in Neural Information Processing Systems*, volume 32, pp. 13771–13781, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Abstract.html.

Yanhao Wang, Francesco Fabbri, and Michael Mathioudakis. Streaming algorithms for diversity maximization with fairness constraints. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 41–53, 2022. doi: 10.1109/ICDE53745.2022.00008.

Jiachen Zhao, Xiao Yan, and Eric Lo. Approximate diverse k-nearest neighbor search in vector database. In *2026 IEEE 42nd International Conference on Data Engineering (ICDE)*, pp. 2971–2983, 2026. doi: 10.1109/ICDE65706.2026.00221.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li. Recommender systems in the era of large language models (LLMs). *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6889–6907, 2024. doi: 10.1109/TKDE.2024.3392335.

Kaiping Zheng, Hongzhi Wang, Zhixin Qi, Jianzhong Li, and Hong Gao. A survey of query result diversification. *Knowledge and Information Systems*, 51(1):1–36, 2017. doi: 10.1007/s10115-016-0990-4.

A NOTATION TABLE

Before introducing our problem setting, we first fix the notation used in this section and throughout the subsequent algorithmic analysis in Table 4. Collecting the recurring symbols in one place makes the notation explicit. Unless otherwise noted, all symbols listed in the table retain the same meaning throughout the paper. Symbols that are specific to a particular proof or algorithm are introduced locally when they first appear.

Table 4: Notation used throughout the paper.

Symbol	Meaning
(X, d)	General metric space and its distance function.
D, $n = \lvert D \rvert$	Finite database and its cardinality.
q	Query object.
K	The specified number of returned data objects.
S, $\mathcal{S}_K(D)$	A selected set and the family of all K-element subsets of D.
$\mathrm{dist}(q, S)$	Average distance from the objects in S to the query object q.
$\mathrm{div}(S)$	Minimum pairwise distance within S.
δ_{max}, Δ	Diameter of the database D.
λ	Parameter to control the trade-off between relevance and diversity.
$f_\lambda(q, S)$	Objective function for our DKNNS formulation.
S^*, $\mathrm{dist}(q, S^*)$, div^*	Optimal result set, its distance to the query q, and its diversity value.
ϵ	Accuracy parameter used by our approximation algorithms.
δ_{min}	Minimum inter-object distance.
$\theta, \theta_j \in \Theta_\epsilon$	The threshold values used by our approximation algorithms.
α	The factor for the certified α-shortcut-reachability assumption introduced in (Indyk & Xu, 2023).

B EXTENDED RELATED WORK

B.1 K-NEAREST NEIGHBOR SEARCH

K-nearest neighbor search (KNNS) in metric spaces has attracted much attention over several decades. Due to the curse of dimensionality, most existing studies focus on approximate KNNS (ANNS). In general, existing ANNS approaches include tree indexes (Bentley, 1975; Muja & Lowe, 2009), locality-sensitive hashing (LSH) (Indyk & Motwani, 1998), quantization (Jégou et al., 2011; Baranchuk et al., 2018; André et al., 2021), and proximity graphs (Malkov et al., 2014; Malkov & Yashunin, 2020; Fu et al., 2019; Subramanya et al., 2019).

Among these approaches, graph-based ANNS approaches attain state-of-the-art empirical performance in the accuracy-efficiency trade-off (Oguri & Matsui, 2023). By organizing database objects as a sparse proximity graph, a greedy or beam search can traverse only a small number of vertices along the edges to obtain ANNS results. Despite their great success, most graph-based ANNS approaches lack worst-case guarantees. To address this, Prokhorenkova & Shekhovtsov (2020) first analyzed graph-based NNS in low dimensions; Indyk & Xu (2023) presented the upper and lower bounds on the worst-case search times of specific proximity graph implementations; Diwan et al. (2024) established the upper and lower bounds on the sparsity required to construct navigable graphs for high-dimensional data; Gollapudi et al. (2025) provided the first worst-case theoretical analysis for the beam search algorithm; Al-Jazzazi et al. (2025) proposed an adaptive beam search algorithm and proved that it returns provable results for KNNS whenever the search graph is navigable. These results provide a foundation for our theoretical analysis of graph-based search methods. However, they focus on ordinary KNNS alone and do not account for diversity.

B.2 DIVERSITY-AWARE SIMILARITY SEARCH

Diversity-aware similarity search aims to find a set of objects that are not only similar to the query but also distinct from one another. This has long been studied in machine learning, information

retrieval, and recommender systems (Drosou & Pitoura, 2010; Santos et al., 2015). A common approach based on KNNS is to first retrieve a set of size K' ($K' \gg K$) of candidate objects with standard ANNS and then rerank them using maximal marginal relevance (MMR) (Carbonell & Goldstein, 1998) or determinantal object process (DPP) (Chen et al., 2018). This approach suffers from a severe efficiency bottleneck since the candidate pool should be very large to ensure a diverse final solution (Anand et al., 2025). Recently, LotusFilter (Matsui, 2025) was proposed to accelerate post-processing with a learned cutoff table, but it still cannot fully escape the bottleneck.

Another line of work adds a diversity term to the distance-based relevance objective or uses it as an additional constraint. Anand et al. (2025) modified graph construction and traversal for ANNS by imposing a diversity constraint that no objects of the same color appear more than a predefined number of times in the solution. Here, the "color" of an object can be assigned based on either attributes or cluster memberships. Barman et al. (2026) designed a welfare-based objective that formulates diversity through Nash and p-mean welfare over object attributes. Both methods, however, rely on categorical attributes for diversity measurement rather than the pairwise distances used in our formulation. Rao et al. (2016) use randomized LSH to trade search accuracy for diversity. Zhao et al. (2026) developed progressive search methods for kNNS with a diversity constraint that limits the distances between any pair of objects to not be below a prespecified threshold.

Beyond KNNS, diversity has also been considered in similarity search problems based on inner product-based relevance measures (Hirata et al., 2022; Huang et al., 2026). These methods, however, cannot be directly applied to diversity-aware KNNS because they are optimized specifically for norm-dependent, non-symmetric inner products.

B.3 Diversity Maximization

Diversity maximization is a classical optimization problem (Drosou & Pitoura, 2010). Among diversity objectives, max-min dispersion (MMD), which finds a set of K objects to maximize the minimum pairwise distance, is one of the most widely studied in the literature. The MMD problem is generally NP-hard for metric distances, and no polynomial-time algorithm can approximate it with a factor better than $1/2$ unless P=NP (Ravi et al., 1994). As such, a broad line of approximation algorithms has been proposed for MMD in various settings and computation models (Aghamolaei et al., 2015; Borassi et al., 2019; Ceccarello et al., 2017; Drosou & Pitoura, 2014; Hassin et al., 1997; Indyk et al., 2014; Moumoulidou et al., 2021; Chandra & Halldórsson, 2001; Amagata, 2023; Wang et al., 2022).

Another common diversity objective is max-sum dispersion (MSD), which maximizes the sum of pairwise distances among the selected objects. Still NP-hard, there have also been many approximation algorithms proposed for MSD (Abbassi et al., 2013; Aghamolaei et al., 2015; Bauckhage et al., 2020; Borassi et al., 2019; Ceccarello et al., 2018; 2017; Cevallos et al., 2016; 2017; Hassin et al., 1997; Indyk et al., 2014; Ravi et al., 1994). We note that max-sum dispersion may still include highly similar objects in the solution. We consequently adopt max-min dispersion as the diversity objective in our formulation.

C Missing Proofs

C.1 Proof of Lemma 2.1

Proof. For every object pair $x, y \in S$, the triangle inequality gives $d(q,x) + d(q,y) \geq d(x,y) \geq \mathrm{div}(S)$. Summing over all $\binom{K}{2}$ pairs and observing that each $d(q,x)$ appears exactly $K-1$ times, we have

$$(K-1)\sum_{x \in S} d(q,x) \geq \binom{K}{2}\mathrm{div}(S).$$

Dividing both sides of the above inequality by $K(K-1)$ gives Lemma 2.1. □

C.2 Proof of Theorem 2.2

Proof. We prove it by reducing any instance of Independent Set (Karp, 1972), a classic NP-hard problem, to an instance of DKNNS in $\mathbb{R}^{n^2}$ in polynomial time.

Given a simple undirected graph $G = (V, E)$ with $V = [n]$ and an integer $2 \leq K \leq n$, the INDEPENDENT SET problem decides whether there exists an independent set (i.e., a set of vertices in V where no two vertices are connected by an edge in E) of size K in G.

The reduction procedure constructs unit vectors whose pairwise distance is smaller on edges in E than on non-edges. First, let us define

$$a = \frac{2n}{n^2 + n - 1}, \quad b = \frac{n^2 - n + 1}{n^2 + n - 1}. \tag{6}$$

It holds that

$$b^2 + (n-1)a^2 = \frac{(n^2 - n + 1)^2 + 4n^2(n-1)}{(n^2 + n - 1)^2} = 1. \tag{7}$$

Let e_i be the i-th standard basis vector in $\mathbb{R}^n$. For each vertex $i \in V$, construct a vector

$$y_i := \left(be_i, \left(z_i^{uv} \right)_{1 \leq u < v \leq n} \right) \in \mathbb{R}^{n^2}, \tag{8}$$

where

$$z_i^{uv} := \begin{cases} a(1,0), & i = u, \\ a(1,0), & i = v \text{ and } (u,v) \in E, \\ a(0,1), & i = v \text{ and } (u,v) \notin E, \\ (0,0), & i \notin \{u, v\}. \end{cases} \tag{9}$$

Every vertex occurs in exactly $n - 1$ pair blocks, each of squared norm a^2. Eq. 7, therefore, implies $\|y_i\|_2^2 = b^2 + (n-1)a^2 = 1$. Moreover, the first n dimensions of y_i, i.e., be_i with $b > 0$, ensure that all vectors y_i ($i = 1, \ldots, n$) are mutually distinct.

Consider two vertices $i < j$. The sub-vectors of the first n dimensions of y_i and y_j are orthogonal, and the only pair block in which both vectors are nonzero is the block associated with $\{i, j\}$. Thus,

$$y_i^\top y_j = \begin{cases} a^2, & (i,j) \in E, \\ 0, & (i,j) \notin E. \end{cases} \tag{10}$$

Since both vectors have unit norm, we have

$$\|y_i - y_j\|_2 = \begin{cases} \sqrt{2 - 2a^2} = d_E, & (i,j) \in E, \\ \sqrt{2} = d_N, & (i,j) \notin E, \end{cases} \tag{11}$$

and $d_E < d_N$. Consequently, for every size-K subset $S \subseteq V$,

$$\operatorname{div}(S) = \begin{cases} d_N, & S \text{ is an independent set of } G, \\ d_E, & S \text{ is not an independent set of } G. \end{cases} \tag{12}$$

The reduced instance of DKNNS consists of a database $D' = \{y_1, \ldots, y_n\}$, a query $q = 0 \in \mathbb{R}^{n^2}$, and a size constraint K. Let δ'_{max} be the diameter of D'. For every $S \in \mathcal{S}_K(D')$,

$$\operatorname{dist}(q, S) = \frac{1}{K} \sum_{y_i \in S} \|y_i\|_2 = 1 \text{ and } f_\lambda(q, S) = (1 - \lambda) - \lambda \operatorname{div}(S) + \delta'_{max}. \tag{13}$$

Since λ and δ'_{max} are independent of S, minimizing $f_\lambda(q, S)$ is equivalent to maximizing $\operatorname{div}(S)$. By Eq. 12, the maximum of $\operatorname{div}(S)$ equals d_N if and only if G has an independent set of size K. Invoking an exact algorithm for the DKNNS instance once and checking at most $\binom{K}{2}$ vertex pairs will consequently decide the instance of INDEPENDENT SET.

The reduction procedure constructs and outputs n vectors in $\mathbb{R}^{n^2}$. Every coordinate of each vector belongs to $\{0, a, b\}$, and the binary encoding lengths of a and b are $O(\log n)$. Thus, the reduction procedure and output length are polynomial. □

C.3 Proof of Theorem 2.3

Proof. Fix $\zeta \in (0, 1/3)$ and $t > 0$. We reduce from CLIQUE (Karp, 1972) to DKNNS. Let (G_0, K) be an instance with $3 \le K \le |V(G_0)|$. Form $G = (V, E)$ by taking the disjoint union of G_0 and one isolated edge. Because the added component has only two vertices and $K \ge 3$, G has a K-clique if and only if G_0 does.

Create one database object for each vertex and a query object q, so $X = V \cup \{q\}$ and $D = V$. Define

$$\forall v \in V,\ d(q, v) = t \qquad \forall u \ne v \in V,\ d(u, v) = \begin{cases} 2t, & (u, v) \in E, \\ t, & (u, v) \notin E \end{cases} \tag{14}$$

and $d(x, x) = 0$. The constructed space (X, d) is a metric space. The identity of indiscernibles, non-negativity, and symmetry of (X, d) are obvious. For any three distinct objects, every side has a length of at most $2t$, whereas the sum of the other two sides is at least $t + t = 2t$, so the triangle inequality holds. Triangles with repeated objects are trivial. Because the added isolated edge has a length of $2t$, the diameter of D is $\delta_{max} = 2t$.

Set $\lambda = 1/2$. Every database object is at a distance t from q, making

$$\mathrm{dist}(q, S) = \frac{1}{K} \sum_{x \in S} t = t \tag{15}$$

for every $S \in \mathcal{S}_K(D)$. Thus, the value of $f_{1/2}(q, S)$ is only affected by $\mathrm{div}(S)$. For any $S \in \mathcal{S}_K(D)$, we have

$$\mathrm{div}(S) = \begin{cases} 2t, & S \text{ corresponds to a } K\text{-clique in } G, \\ t, & \text{otherwise.} \end{cases} \tag{16}$$

Accordingly, we have

$$f_{1/2}(q, S) = \begin{cases} \dfrac{3t}{2} = L, & S \text{ corresponds to a } K\text{-clique in } G, \\ 2t = H, & \text{otherwise.} \end{cases} \tag{17}$$

If (G_0, K) is a YES instance, a K-clique attains L, the smaller of the two possible values, so $\mathrm{OPT} = L$. If (G_0, K) is a NO instance, G has no K-clique, and every feasible set has a value of H, so $\mathrm{OPT} = H$. Obviously, $\frac{H}{L} = \frac{4}{3}$.

Suppose that a polynomial-time α-approximation algorithm $\mathcal{A}$ exists for $\alpha = 4/3 - \zeta$. Given (G_0, K), construct the instance above, run $\mathcal{A}$, and let $S_{\mathcal{A}}$ be its output. On a YES instance,

$$f_{1/2}(q, S_{\mathcal{A}}) \le \left(\frac{4}{3} - \zeta\right) L < H.$$

By Eq. 17, the output cannot be a non-clique set of value H; it must correspond to a K-clique. On a NO instance, no K-clique exists, so the output necessarily fails the clique check. Therefore, checking whether the vertices returned by $\mathcal{A}$ form a clique would decide CLIQUE in polynomial time, contradicting $P \ne NP$. □

C.4 Proof of Lemma 3.1

Proof. For each previously selected object p_j, consider the open ball

$$B(p_j, \theta) = \{x \in D : d(x, p_j) < \theta\}. \tag{18}$$

Indeed, if $u, v \in S^*$ both belonged to $B(p_j, \theta)$, the triangle inequality would give

$$d(u, v) \le d(u, p_j) + d(p_j, v) < 2\theta \le \mathrm{div}^*.$$

This contradicts the definition of div^*.

At step i, an object of S^* is excluded from F_i only if it lies in one of the $i - 1$ balls centered at $p_1, \ldots, p_{i-1}$. Since each ball contains at most one object in S^*, at least $K - i + 1$ objects from S^* remain in F_i. Therefore, F_i is nonempty for every $i \le K$.

Next, we construct a matching to S^* in reverse order. Starting from $i = K$ and proceeding down to $i = 1$, choose

$$s_i \in (S^* \cap F_i) \setminus \{s_{i+1}, \ldots, s_K\}.$$

At most $K - i$ objects in S^* have already been used, whereas $|S^* \cap F_i| \geq K - i + 1$; thus, such a choice always exists. The objects $s_1, \ldots, s_K$ are therefore distinct and exhaust S^*. Applying $d(q, p_i) \leq \beta \min_{z \in F_i} d(q, z)$ to $s_i \in F_i$ yields

$$d(q, p_i) \leq \beta d(q, s_i).$$

Summing over i and dividing by K gives

$$\operatorname{dist}(q, T) = \frac{1}{K} \sum_{i=1}^{K} d(q, p_i) \leq \frac{\beta}{K} \sum_{i=1}^{K} d(q, s_i) = \beta \operatorname{dist}(q, S^*).$$

Finally, each p_i is feasible w.r.t. all earlier selections. Thus, every pair of objects in T is at a distance of at least θ. □

C.5 Proof of Lemma 3.2

Proof. Since $\theta_0 = \delta_{min}/2 \leq \operatorname{div}^* /2 \leq \delta_{max}/2 = \theta_J$, the threshold $\widehat{\theta}$ must exist. Write $\widehat{\theta} = \theta_j$.

If $j < J$, the maximality of $\widehat{\theta}$ implies

$$\theta_j \leq \frac{\operatorname{div}^*}{2} < \theta_{j+1} = \theta_j + h_\epsilon.$$

Therefore,

$$\widehat{\theta} > \frac{\operatorname{div}^*}{2} - h_\epsilon.$$

If $j = J$, then $\theta_J \leq \operatorname{div}^* /2 \leq \theta_J$, and hence $\widehat{\theta} = \operatorname{div}^* /2$. Together with $\widehat{\theta} \geq \theta_0$, this proves

$$\max\left\{\frac{\delta_{min}}{2}, \frac{\operatorname{div}^*}{2} - h_\epsilon\right\} \leq \widehat{\theta} \leq \frac{\operatorname{div}^*}{2}.$$

□

C.6 Proof of Theorem 3.3

Proof. First, if $\delta_{min} = \delta_{max}$, every feasible set has the same diversity, and TGS returns an optimal solution. Assume henceforth that $0 < \delta_{min} < \delta_{max}$.

Since $J = \left\lceil \frac{1}{\epsilon} \right\rceil$ and thus $\frac{1}{J} \leq \epsilon$, together with $\delta_{max} - \delta_{min} < \delta_{max}$, we have

$$h_\epsilon = \frac{\delta_{max} - \delta_{min}}{2J} \leq \frac{\epsilon \delta_{max}}{2}.$$

By Lemmas 3.1 and 3.2, the candidate $T_{\widehat{\theta}}$ satisfies

$$\operatorname{dist}(q, T_{\widehat{\theta}}) \leq \operatorname{dist}(q, S^*), \qquad \operatorname{div}(T_{\widehat{\theta}}) \geq \operatorname{div}^* - \frac{(1+\epsilon)\delta_{max}}{2}. \tag{19}$$

Because TGS returns the candidate with the minimum objective value,

$$\begin{aligned} f_\lambda(q, S_{\mathrm{TGS}}) &\leq f_\lambda(q, T_{\widehat{\theta}}) \\ &\leq (1-\lambda) \operatorname{dist}(q, S^*) - \lambda \left(\operatorname{div}^* - \frac{(1+\epsilon)\delta_{max}}{2} \right) + \delta_{max} \\ &= f_\lambda(q, S^*) + \frac{\lambda(1+\epsilon)\delta_{max}}{2}. \end{aligned} \tag{20}$$

By Lemma 2.1,

$$\operatorname{dist}(q, S^*) \geq \frac{\operatorname{div}^*}{2}.$$

Therefore,

$$\begin{aligned} f_\lambda(q, S^*) &= (1-\lambda)\,\mathrm{dist}(q, S^*) + \delta_{max} - \lambda\,\mathrm{div}^* \\ &\geq \delta_{max} + \frac{1-3\lambda}{2}\,\mathrm{div}^* \\ &\geq \begin{cases} \delta_{max}, & 0 \leq \lambda \leq 1/3, \\ \dfrac{3(1-\lambda)\delta_{max}}{2}, & 1/3 \leq \lambda \leq 1/2. \end{cases} \end{aligned} \tag{21}$$

Dividing Eq. 20 by the corresponding lower bound gives

$$\frac{f_\lambda(q, S_{\mathrm{TGS}})}{f_\lambda(q, S^*)} \leq \rho_\epsilon(\lambda) := \begin{cases} 1 + \dfrac{\lambda(1+\epsilon)}{2}, & 0 \leq \lambda \leq 1/3, \\ 1 + \dfrac{\lambda(1+\epsilon)}{3(1-\lambda)}, & 1/3 \leq \lambda \leq 1/2. \end{cases} \tag{22}$$

Since $\lambda \leq 1/2$,

$$\frac{f_\lambda(q, S_{\mathrm{TGS}})}{f_\lambda(q, S^*)} \leq \rho_\epsilon(\lambda) \leq \rho_\epsilon(1/2) = \frac{4}{3} + \frac{\epsilon}{3} \leq \frac{4}{3} + \epsilon.$$

This completes the proof. □

C.7 Complexity Analysis of TGS

First, sorting all objects in D by their distance to q takes $O(n \log n)$ comparisons. For each indexed threshold $\theta_j \in \Theta_\epsilon$, the scan examines the sorted database until either K objects have been accepted or the database is exhausted. Since each acceptance test requires at most $K-1$ distance evaluations and the grid contains $J+1 = O(1/\epsilon)$ thresholds, the solution computation takes

$$O(nK/\epsilon)$$

distance evaluations.

Evaluating the objective value for all candidates adds at most $O(|\Theta_\epsilon|K^2)$ distance evaluations. Because $K \leq n$, this term is absorbed by the preceding bound. Therefore, assuming constant-time distance evaluations and sorting comparisons, the total time complexity of TGS is

$$O(n\,(\log n + K/\epsilon))\,.$$

C.8 Proof of Theorem 4.1

Proof. We first show that the traversal eventually expands every object in D. Fix any current object x and target object z. By Eq. 5, either $(x, z) \in E$ or there exists $y \in N^+(x)$ such that

$$d(y, z) \leq \frac{d(x, z)}{\alpha} < d(x, z),$$

where the strict inequality follows from $\alpha > 1$. Repeating this step strictly decreases the distance to z. Since D is finite, the process eventually reaches z. Thus, every object in D is reachable from the entry object s. Because the unbounded priority queue does not discard discovered objects, MTGS eventually expands all objects in D.

Let $\widehat{\theta}$ be the largest threshold in Θ_ϵ satisfying $\widehat{\theta} \leq \mathrm{div}(S^*)/2$. By Lemma 3.2,

$$\frac{\mathrm{div}(S^*)}{2} - h_\epsilon \leq \widehat{\theta} \leq \frac{\mathrm{div}(S^*)}{2},$$

where $h_\epsilon \leq \frac{\epsilon\Delta}{2}$.

We next show that MTGS completes $T_{\widehat{\theta}}$. Suppose that fewer than K objects have been selected after all objects in D have been processed. Every object $u \in S^*$ is then either selected or rejected because some selected object $p \in T_{\widehat{\theta}}$ satisfies $d(u, p) < \widehat{\theta}$.

The open ball of radius $\widehat{\theta}$ centered at a selected object cannot contain two distinct objects $u, v \in S^*$. Otherwise,

$$d(u, v) \le d(u, p) + d(p, v) < 2\widehat{\theta} \le \operatorname{div}(S^*),$$

contradicting the definition of $\operatorname{div}(S^*)$. Hence, each selected object can account for at most one object of S^*. Fewer than K selected objects therefore cannot account for all K objects in S^*. Consequently, $|T_{\widehat{\theta}}| = K$.

By the acceptance rule and the critical-threshold bound,

$$\operatorname{div}(T_{\widehat{\theta}}) \ge \widehat{\theta} \ge \frac{\operatorname{div}(S^*)}{2} - \frac{\epsilon\Delta}{2} \ge \operatorname{div}(S^*) - \frac{(1+\epsilon)\Delta}{2}, \tag{23}$$

where the last inequality follows from $\operatorname{div}(S^*) \le \Delta$.

For every $x \in T_{\widehat{\theta}}$ and $z \in S^*$, the triangle inequality gives

$$d(q, x) \le d(q, z) + d(z, x) \le d(q, z) + \Delta.$$

Averaging first over $z \in S^*$ and then over $x \in T_{\widehat{\theta}}$ yields

$$\operatorname{dist}(q, T_{\widehat{\theta}}) \le \operatorname{dist}(q, S^*) + \Delta. \tag{24}$$

Since MTGS returns the minimum-objective set among all complete threshold candidates,

$$\begin{aligned} f_\lambda(q, S_{\mathrm{MTGS}}) &\le f_\lambda(q, T_{\widehat{\theta}}) \\ &\le (1-\lambda)(\operatorname{dist}(q, S^*) + \Delta) - \lambda\left(\operatorname{div}(S^*) - \frac{(1+\epsilon)\Delta}{2}\right) + \Delta \\ &= f_\lambda(q, S^*) + A_\epsilon(\lambda)\Delta, \end{aligned} \tag{25}$$

where

$$A_\epsilon(\lambda) := 1 - \lambda + \frac{\lambda(1+\epsilon)}{2} = 1 - \frac{\lambda(1-\epsilon)}{2}.$$

By Lemma 2.1, $\operatorname{dist}(q, S^*) \ge \operatorname{div}(S^*)/2$. Therefore,

$$\begin{aligned} f_\lambda(q, S^*) &= (1-\lambda)\operatorname{dist}(q, S^*) - \lambda \operatorname{div}(S^*) + \Delta \\ &\ge \Delta + \frac{1-3\lambda}{2}\operatorname{div}(S^*) \\ &\ge \begin{cases} \Delta, & 0 \le \lambda \le 1/3, \\ \dfrac{3(1-\lambda)\Delta}{2}, & 1/3 \le \lambda \le 1/2. \end{cases} \end{aligned} \tag{26}$$

If $0 \le \lambda \le 1/3$, then $A_\epsilon(\lambda) \le 1$. Combining Eqs. 25 and 26 gives

$$\frac{f_\lambda(q, S_{\mathrm{MTGS}})}{f_\lambda(q, S^*)} \le 1 + A_\epsilon(\lambda) \le 2.$$

If $1/3 \le \lambda \le 1/2$, the same equations give

$$\frac{f_\lambda(q, S_{\mathrm{MTGS}})}{f_\lambda(q, S^*)} \le 1 + \frac{2A_\epsilon(\lambda)}{3(1-\lambda)} = 1 + \frac{2 - \lambda + \lambda\epsilon}{3(1-\lambda)} \le 2 + \frac{\epsilon}{3} \le 2 + \epsilon.$$

The inequality follows from $(1+\epsilon)(1-2\lambda) \ge 0$.

Combining the two cases proves

$$\frac{f_\lambda(q, S_{\mathrm{MTGS}})}{f_\lambda(q, S^*)} \le 2 + \epsilon.$$

□

C.9 Complexity Analysis of MTGS

We analyze the time complexity of MTGS. Let $M := |D_{\text{seen}}|$ at termination, let V_{exp} denote the set of expanded objects, and define

$$E_{\text{exp}} := \sum_{v \in V_{\text{exp}}} |N^+(v)|.$$

With hash-based duplicate detection, scanning the outgoing edges of all expanded objects takes $O(E_{\text{exp}})$ operations. Each discovered object requires one query-distance evaluation and is inserted into and removed from the shared priority queue at most once. Hence, the shared best-first graph traversal requires $O(E_{\text{exp}} + M \log M)$ operations.

For each threshold $\theta \in \Theta_\epsilon$, MTGS maintains only the selected set T_θ and does not construct a separate search queue. Whenever an object is expanded, it is passed to every incomplete threshold state. Testing whether the object can be inserted into T_θ requires comparisons with at most K previously selected objects. Therefore, the threshold-specific work over all expanded objects is $O(|\Theta_\epsilon| MK)$ distance evaluations. Since $|\Theta_\epsilon| = J + 1 = O(1/\epsilon)$, the total query-time complexity is

$$O(E_{\text{exp}} + M \log M + (J+1)MK) = O\big(E_{\text{exp}} + M \log M + MK\epsilon^{-1}\big). \tag{27}$$

Evaluating the objective values of all completed candidates requires at most $O((J+1)K^2)$ pairwise distance evaluations. This term is absorbed by the threshold-specific bound because $M \geq K$ whenever a complete candidate exists.

Under the α-shortcut-reachability assumption, every object is reachable from the entry object by a directed path. Since MTGS never discards a discovered object and terminates only after the shared priority queue becomes empty, induction along these paths shows that every object is eventually discovered and expanded. Consequently, $M = n$ and $E_{\text{exp}} = |E|$ yield the worst-case bound

$$O\big(|E| + n \log n + nK\epsilon^{-1}\big).$$

D Supplementary Experiments

D.1 Effect of Error Parameter

We study the effect of the threshold-grid accuracy parameter by varying ϵ for TGS and MTGS on GIST while keeping all other settings fixed. External baselines are omitted because ϵ is specific to our threshold-grid construction. The results are shown in Fig. 3.

Both methods are relatively insensitive to ϵ in the fine-grained range. For $\epsilon \leq 0.05$, the objective ranges from 0.7139 to 0.7186 for TGS and from 0.7191 to 0.7278 for MTGS. Increasing ϵ from 0.001 to the default 0.005 changes the objective by only 0.0006 and 0.0008, respectively, while reducing query time from $29{,}661.3$ to $8{,}080.4$ ms for TGS and from 127.0 to 49.3 ms for MTGS. These correspond to reductions of 72.8% and 61.2%. At the default setting, MTGS is approximately $164\times$ faster than TGS.

For larger ϵ, the query-time curves gradually flatten, whereas the objective continues to increase, particularly beyond 0.2. Thus, $\epsilon = 0.005$ provides a favorable balance between objective quality and query efficiency. The observed trend is also consistent with the $O(1/\epsilon)$ threshold-grid dependence in our analysis.

D.2 Scalability Evaluation on Synthetic Data

We further evaluate TGS and MTGS on synthetic Gaussian-mixture datasets by varying cluster density, database size, and dimension. The results are shown in Figure 4.

Database Size The objective value decreases similarly for both methods as the database grows since denser sampling provides more opportunities to find points that are both close to the query and mutually separated. Their query time trends differ substantially: TGS grows rapidly with the full-scan workload, whereas the bounded graph traversal of MTGS is much less affected by database size, yielding a $7.7\times$ speedup at 100k points.

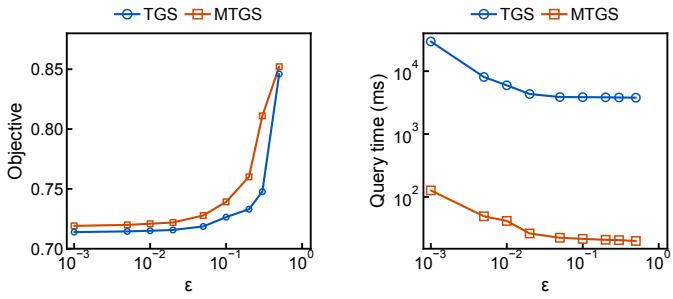


Figure 3: Effect of the value of ϵ on the performance of TGS and MTGS.

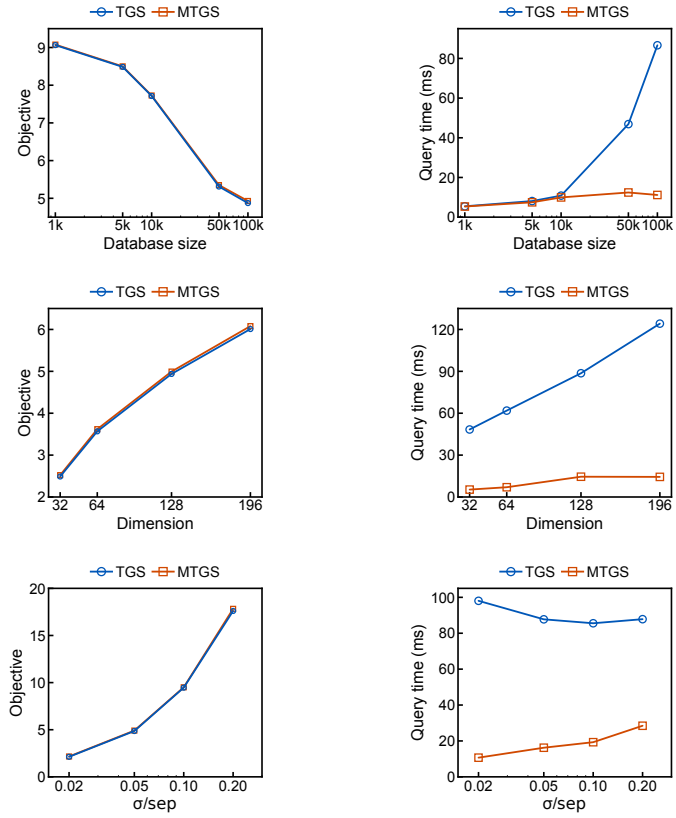


Figure 4: Synthetic-data sensitivity studies across database scale, vector dimension, and density.

Dimension Both objective values and query time increase with the dimension, reflecting the changing scale of Euclidean distances and the higher cost of each distance evaluation. Nevertheless, the objective of MTGS remains close to that of TGS, while MTGS is $8.6\times$ faster at 196 dimensions.

Cluster Density The objective value increases steadily with cluster density, indicating that stronger cluster overlap makes it harder to balance query relevance and result diversity. The query time of TGS remains nearly constant because it always scans the full dataset. MTGS requires more local distance evaluations as neighborhoods become denser, but remains $3.1\times$ faster than TGS even in the very-dense setting.