Canopy-Guided Construction of ANN Search Graphs under Cosine Similarity

Rafael F. Pinheiro[1], **Karla Roberta P. S. Lima**[1]

[1]PPGSI-EACH
Universidade de São Paulo (USP)

{r.pinheiro,ksampaiolima}@usp.br

Abstract. ***Research Context:*** *Nearest Neighbor Search (NNS) is a fundamental problem in computing, with applications in recommendation systems, semantic search, information retrieval, and Retrieval-Augmented Generation (RAG) in Large Language Models (LLMs). High-dimensional datasets with millions of vectors make exact search computationally intractable, demanding efficient approximations.* ***Practical Problem:*** *Approximate Nearest Neighbor Search (ANNS) has become the standard approach, with graph-based methods standing out. However, structures such as Hierarchical Navigable Small World (HNSW) graphs impose high construction and update costs, which limits their scalability in real-world scenarios.* ***Proposed Solution:*** *This paper proposes the integration of Canopy Clustering as a divide-and-conquer heuristic to mitigate the computational burden of HNSW graph construction under cosine similarity, enabling more cost-effective ANNS pipelines in information retrieval systems.* ***Related IS Theory:*** *The research aligns with Information Systems theories on efficiency of knowledge retrieval, and the optimization of data-intensive processes in decision-support environments. It builds upon principles of data organization and access structures in IS, particularly the trade-offs between accuracy and computational feasibility in large-scale information retrieval.* ***Research Method:*** *Controlled experiments were conducted to evaluate the applicability of divide-and-conquer and Canopy Clustering prior to HNSW construction. Comparative analyses measured build time, query latency, and recall performance.* ***Summary of Results:*** *Preliminary findings suggest that Canopy Clustering can reduce construction costs while sustaining recall levels adequate for IS applications.* ***Contributions and Impact to IS area:*** *The work introduces a novel heuristic combination to enhance the scalability of graph-based similarity search indexing, providing methodological advances that may extend to other high-dimensional data processing tasks. It offers actionable insights for designing more efficient and viable intelligent information systems.*

1. Introduction

Recommendation systems, information retrieval, pattern recognition, and, more recently, semantic search based on embeddings from large language models (LLMs) [Reimers and Gurevych 2019] for prompt engineering purposes [Brown et al. 2020] are among the main applications of Nearest Neighbor Search (NNS), a fundamental problem in computer science that consists of finding the most similar data points to a given query.

As data-driven applications increasingly rely on vector embeddings to represent users, documents, images, and other entities, the scale of modern datasets has indeed expanded to millions—or even billions—of vectors with hundreds or thousands of dimensions. However, in such contexts, the exact search becomes computationally prohibitive [Wang et al. 2021], as it requires exhaustive pairwise distance computations that lead to unrealistic time and memory expenditures, making it impractical for real-world Information Systems that demand low-latency responses.

Approximate Nearest Neighbor Search (ANNS) techniques have emerged as a practical alternative by sacrificing a fraction of accuracy in order to achieve significant improvements in efficiency. However, the most successful ANNS approaches to date — particularly graph-based methods such as HNSW — introduce new challenges. While they deliver state-of-the-art recall-speed trade-offs at query time, their construction and update costs remain high, often scaling with accelerated growth as dataset size and graph parameters increase. This limitation poses a critical barrier for scenarios in which embeddings are frequently updated or system scalability is essential.

From a practical standpoint, this bottleneck directly impacts core Information Systems applications, such as recommendation engines, retrieval-augmented generation (RAG) [Lewis et al. 2020] pipelines, and knowledge management platforms, all of which rely on the ability to search vast embedding spaces quickly and accurately. As embeddings become ubiquitous in IS infrastructures, the need for methods that combine scalability with computational efficiency grows increasingly urgent. Beyond the technological dimension, this challenge also affects organizational processes and people: high construction costs translate into delayed updates and limited responsiveness of IS applications, which, in turn, reduce user satisfaction and constrain decision-making agility.

A divide-and-conquer pipeline is proposed, in which Canopy Clustering precedes graph construction for Approximate Nearest Neighbor Search under cosine similarity. The central idea is to replace a single, monolithic ANNS index over all vectors with a set of partially overlapping subproblems (“canopies”), each small enough to lower construction and update costs without materially degrading recall. To ground and evaluate this proposal, Section 2 provides the theoretical background on ANNS, HNSW, and Canopy Clustering; Section 3 reviews related work; Section 4 describes the materials, methods, and proposed algorithmic variants; Section 5 presents the experimental results; and Section 6 concludes with contributions and directions for future research.

2. Background

The proposed approach is grounded in the intersection of algorithmic advances in Approximate Nearest Neighbor Search (ANNS) and sociotechnical perspectives in Information Systems (IS). On the algorithmic side, the research builds upon decades of work on divide-and-conquer strategies, in which large problems are decomposed into smaller, more tractable subproblems and then recombined. Classical paradigms such as Merge Sort or binary search epitomize this principle, and their success has inspired modern adaptations in data-intensive domains. In the context of ANNS, this strategy offers a natural way to tackle the computational bottlenecks inherent in high-dimensional vector search by partitioning the problem into smaller subspaces while retaining enough connectivity to approximate global structure.

Within this framework, Canopy Clustering [McCallum et al. 2000] emerges as an appealing candidate for preprocessing. Originally introduced as a lightweight technique for creating overlapping clusters using dual thresholds, Canopy was often employed as a preliminary step to accelerate traditional clustering algorithms such as K-means. Its strengths—low computational cost, flexibility of overlap, and robustness to noisy distances—make it particularly suitable for tasks in which exhaustive pairwise comparisons are prohibitive. Despite these advantages, its potential within ANNS pipelines remains underexplored: existing studies have not systematically investigated how Canopy's overlapping partitions might reduce graph-construction costs while preserving search quality.

The current state of the art in ANNS is dominated by graph-based methods, most notably Hierarchical Navigable Small World (HNSW) [Malkov and Yashunin 2018], which has been extensively studied and benchmarked [Aumüller et al. 2020]. HNSW achieves excellent recall-speed trade-offs through layered proximity graphs, but its construction phase is costly, especially at scale, and the algorithm does not natively exploit potential data structure revealed by clustering. Alternative approaches such as KD-trees or Locality Sensitive Hashing (LSH) have historically played a role but degrade severely under high dimensionality, underscoring the need for hybrid heuristics that exploit both structural decomposition and graph navigability.

From the IS perspective, this research emphasizes scalability and innovation in intelligent systems. Efficient search in high-dimensional embedding spaces is no longer an isolated computer science challenge but a core enabler of IS applications, including recommender systems, knowledge management platforms, and Retrieval-Augmented Generation pipelines. By proposing a canopy-enhanced ANNS pipeline, the study contributes to these sociotechnical concerns by showing how algorithmic heuristics can translate into practical improvements in IS performance. Beyond the technological advance, reducing construction costs makes large-scale, real-time deployment more viable for organizations while simultaneously improving responsiveness for users. This aligns with the GranDSI-BR 2016–2026 [Boscarioli et al. 2017] research agenda, particularly the challenge of adopting a sociotechnical view of Information Systems, which emphasizes not only computational efficiency but also the integration of people, processes, and technology in real-world contexts.

3. Related Work

In the field of ANNS, a wide variety of methods have been proposed to mitigate the limitations of exact search in high-dimensional spaces. Classical approaches based on trees or hashing offered early solutions but faced significant degradation as dimensionality increased. More recently, graph-based methods such as HNSW have become the state of the art, combining high recall with efficient query times, though at considerable construction costs. In parallel, lightweight clustering heuristics like Canopy Clustering have been explored mainly as preprocessing tools in data mining and machine learning, but their role within ANNS pipelines remains largely unexplored. This section reviews the main strands of related literature — ranging from general NNS formulations and graph-based ANNS to clustering heuristics — highlighting the existing gaps and the space for hybrid approaches such as the one proposed in this study.

3.1. Nearest Neighbor Search

The Nearest Neighbor Search (NNS) problem consists of identifying, within a dataset, the elements most similar to a given query point according to a defined distance or similarity measure.

Two widely used distance functions in NNS are Euclidean distance and cosine distance, the latter corresponding to subtracting cosine similarity from 1. Euclidean distance is sensitive to vector magnitude and is typically employed in geometrically interpretable spaces. In contrast, cosine distance is direction-based and ignores magnitude, making it particularly suitable for high-dimensional, sparse data such as text embeddings or word vectors. When vectors are normalized, cosine distance becomes functionally equivalent to a transformation of Euclidean distance, enabling hybrid or interchangeable use in some computational settings.

Formally, given a query vector $q \in \mathbb{R}^d$ and a dataset $S = \{s_1, s_2, ..., s_n\} \subset \mathbb{R}^d$, the goal is to retrieve a subset $R \subseteq S$, typically the top-k closest elements to q based on a distance function δ, such that:

$$R = \arg \min_{R \subseteq S, |R|=k} \sum_{x \in R} \delta(x, q)$$

Alternatively, results can be filtered by a distance threshold τ, returning only those elements satisfying $\delta(q, p_i) \leq \tau$. Notably, due to the high computational cost of exact NNS in large and high-dimensional datasets, Approximate Nearest Neighbor Search (ANNS) has emerged as a practical alternative [Aguerrebere et al. 2023].

Moreover, the behavior of cosine distance in high-dimensional spaces must be carefully considered as it tends to converge towards 1 as dimensionality increases, causing, for example, similarity scores between random vectors to become nearly indistinguishable. This phenomenon, a manifestation of the curse of dimensionality, narrows the effective dynamic range of similarity scores and complicates the selection of distance thresholds for search or clustering algorithms.

3.2. Approximate Nearest Neighbor Search (ANNS)

Graph-based approaches to ANNS have emerged as one of the most effective solutions for high-dimensional vector search. Graph-based indices explicitly model neighborhood relationships among data points, enabling efficient navigation during query time. In these methods, the dataset is represented as a graph in which vertices correspond to vectors and edges connect each vertex to a subset of its nearest neighbors.

During query processing, the search starts from one or more entry points in the graph and iteratively explores local neighborhoods using greedy strategies. The traversal moves toward vertices that are progressively closer to the query vector until convergence to a local optimum is reached. This strategy allows graph-based methods to substantially reduce the number of distance computations compared to exhaustive search, while preserving high recall and precision rates.

Several graph structures have been proposed to optimize both construction and query efficiency. Early approaches explored proximity graphs such as the Delaunay graph

or the k-nearest neighbor (k-NN) graph, but these were often impractical due to high computational costs in large-scale settings. A more advanced solution, Hierarchical Navigable Small World (HNSW) graph introduced hierarchical and navigability properties that significantly improved scalability and search quality.

3.3. Hierarchical Navigable Small World (HNSW)

HNSW graph [Malkov and Yashunin 2018] represents a significant advancement in ANNS, particularly in high-dimensional spaces. HNSW builds upon the Navigable Small World (NSW) framework [Malkov et al. 2014], itself inspired by classical proximity graph structures such as the Delaunay Graph (DG) [Fortune 1995] and the Relative Neighborhood Graph (RNG) [Toussaint 1980]. In RNG, two points are connected only if no third point lies closer to either of them, enforcing a spatial distribution of neighbors that improves search efficiency. Similarly, the DG ensures exact neighbor retrieval while avoiding redundant edges, but its performance deteriorates with increasing dimensionality due to the rapid growth of vertex connections.

NSW was proposed to address some of these limitations by combining long-range and short-range connections in an undirected graph, thereby maintaining global connectivity and supporting efficient greedy search traversals. However, NSW suffers from polylogarithmic search complexity and reduced efficiency in densely populated regions, in which high vertex degrees hinder traversal.

HNSW extends NSW through a hierarchical, multilayered graph organization in which vertices are distributed across multiple levels, each level containing a limited set of neighbors. Search begins in sparse upper layers and progressively descends to denser lower layers, refining candidate neighbors until the closest match is found. This design reduces search complexity to logarithmic time while balancing efficiency and accuracy. Nevertheless, the hierarchical structure introduces higher memory requirements, which can limit scalability for very large datasets. Furthermore, as dimensionality increases, the advantages of the hierarchical layers tend to diminish [Wang et al. 2021].

3.4. Canopy Clustering

Canopy Clustering [McCallum et al. 2000], is a pre-clustering technique designed to reduce the computational cost of more intensive clustering algorithms on high-dimensional and large-scale datasets. This method is particularly suitable as a preliminary step in divide-and-conquer strategies, in which an expensive problem is partitioned into simpler subproblems to improve overall performance.

The algorithm operates in two main stages. First, it uses a fast, approximate distance metric — typically Euclidean distance with two thresholds, T_1 and T_2, where $T_1 > T_2$ — to group data points into overlapping subsets known as canopies. Points within distance T_2 of a selected center are added to the core part of a canopy and removed from further consideration, while those within T_1 remain in the dataset and may be included in multiple canopies. In the second stage, a more accurate measure or other computationally expensive processing is applied exclusively within these canopies, which significantly reduces the number of pairwise comparisons.

This technique has been applied in contexts such as bibliographic citation clustering and text mining, in which data is represented in high-dimensional vector spaces.

By limiting the scope of expensive similarity computations to smaller, relevant subsets, Canopy Clustering enables scalable and parallelizable solutions.

Despite its advantages, the effectiveness of Canopy Clustering depends on carefully selecting the thresholds T_1 and T_2, particularly in high-dimensional spaces affected by the "curse of dimensionality", which can cause similarity measures like cosine distance to concentrate within a narrow range. Additionally, to preserve clustering quality, the canopy assignment must ensure full coverage of the true clusters, either by containing all points in a traditional cluster within a single canopy or ensuring transitive connectivity through overlapping canopies.

This work has explored the use of Canopy Clustering beyond traditional clustering scenarios, as a preprocessing stage in ANNS frameworks based on graph construction. In such cases, canopies may help restrict the graph-building process to smaller and coherent subsets, improving both construction time and query performance.

4. Materials and Methods

To evaluate the proposed integration of Canopy Clustering with graph-based Approximate Nearest Neighbor Search, a set of controlled experiments was conducted to compare different algorithmic variants under consistent conditions. This section details the datasets employed, the experimental setup, and the evaluation metrics, followed by the description of the algorithms under study. In addition, the theoretical complexity of each variant is discussed, along with a comparative analysis of expected trade-offs and conceptual illustrations highlighting their structural differences. Together, these elements provide the methodological foundation for assessing both the scalability and retrieval quality of the proposed approach.

4.1. Datasets and Experimental Setup

The experiments were conducted using collections of pre-normalized, randomly generated vectors with 1,000 dimensions and varying dataset sizes, in order to assess the scalability and robustness of the algorithms. All trials were executed in a controlled environment, with homogeneous hardware and standardized software configurations, to ensure comparability across runs.

A grid search over parameter combinations was employed, defined as:

- $\boldsymbol{M}$: maximum number of connections per node per layer [4, 8, 16, 32].
- $\boldsymbol{efConstruction}$: size of the dynamic candidate list during index construction [10, 50, 100].
- $\boldsymbol{efSearch}$: size of the dynamic candidate list during query [20, 50, 100].
- $\boldsymbol{S}$: number of vectors to index [500, 800, 1,300, 2,100, 3,400, 5,500, 8,900, 14,400, 23,300].

For each algorithm evaluated, 324 parameter settings were tested, and three metrics were collected for each configuration:

1. **Recall@100**: fraction of the 100 exact nearest neighbors that are included in the top 100 retrieved.
2. **Indexing time (build time)**: total time to construct the index.
3. **Query time**: mean runtime for five k-NN search queries (k=100).

4.2. Algorithms Evaluated

4.2.1. Baseline HNSW

The baseline HNSW implementation follows the original formulation [Malkov and Yashunin 2018], as described in Algorithms 1, 2 and 3.

Algorithm 1 Baseline HNSW Insertion [Malkov and Yashunin 2018]

function INSERT_HNSW(graph, q)
  $l \leftarrow sample\_level(assign\_probas)$
  $ep \leftarrow entry\_point$
  for $layer = L$ **down to** $l + 1$ **do**
    $ep \leftarrow search\_layer(q, ep, ef = 1, layer)$
  for $layer = \min(L, l)$ **down to** 0 **do**
    $candidates \leftarrow search\_layer(q, ep, ef = efConstruction, layer)$
    $neighbors \leftarrow select\_neighbors(candidates, M)$
    $add\_bidirectional\_connections(q, neighbors, layer)$
  $update\_entry\_point(q, l)$

Algorithm 2 Baseline HNSW Search Layer [Malkov and Yashunin 2018]

function SEARCH_LAYER(q, ep, ef, layer)
  $V \leftarrow \{ep\}$
  $C \leftarrow \{ep\}$
  $W \leftarrow \{ep\}$
  while $|C| > 0$ **do**
    $c \leftarrow extract\_closest(C, q)$
    $f \leftarrow get\_furthest(W, q)$
    if $distance(c, q) > distance(f, q)$ **then**
      break
    $N \leftarrow neighborhood(c, layer)$
    for all $n \in N$ **do**
      if $n \notin V$ **then**
        $V \leftarrow V \cup \{n\}$
        $f \leftarrow get\_furthest(W, q)$
        if $distance(n, q) < distance(f, q)$ **or** $|W| < ef$ **then**
          $C \leftarrow C \cup \{n\}$
          $W \leftarrow W \cup \{n\}$
          if $|W| > ef$ **then**
            $remove(W, f)$
  return W

Algorithm 3 Baseline HNSW Query [Malkov and Yashunin 2018]

function SEARCH_HNSW(graph, q, k, efSearch)
    $ep \leftarrow entry\_point$
    for $layer = L$ **down to** 1 **do**
        $ep \leftarrow search\_layer(q, ep, ef = 1, layer)$
    $W \leftarrow search\_layer(q, ep, ef = efSearch, layer = 0)$
    return $top\_k(W, k)$

These baseline algorithms serve as the point of comparison for the proposed canopy-based variants. They highlight both the efficiency of HNSW during query time and the computational costs incurred during index construction.

4.2.2. Canopy-Based Multi-Graph HNSW

In this variant, vectors are first assigned to independent canopies, each of which maintains its own HNSW subgraph. Algorithm 4 illustrates the insertion process, while Algorithm 5 describes how queries can be processed across multiple subgraphs.

Algorithm 4 Canopy-Based Multi-Graph HNSW Insertion [Author's research]

function MULTI_INSERT(canopies, q)
    $assign\_canopies \leftarrow compatible\_canopies(q)$
    for all $c \in assign\_canopies$ **do**
        $insert\_hnsw(canopies[c], q)$

Algorithm 5 Canopy-Based Multi-Graph HNSW Query [Author's research]

function MULTI_SEARCH(canopies, q, k, efSearch)
    $W \leftarrow \{\}$
    for all $c \in canopies$ **do**
        $partial \leftarrow search\_HNSW(canopies[c], q, k, efSearch)$
        $W \leftarrow W \cup \{partial\}$
    return $top\_k(W, k)$

This design favors scalability and parallelism, but introduces the challenge of merging results from multiple subgraphs. As such, it provides a useful middle ground between the Baseline HNSW and the more elaborate integrated structure presented next.

4.2.3. Bridged-HNSW

The Bridged-HNSW approach extends the Canopy-Based Multi-Graph HNSW variant by introducing cross-canopy connectivity through shared elements and global seeds. Algorithm 6 presents the insertion procedure, and Algorithms 7, 8 and 9 illustrate how queries traverse the integrated structure still based on multiple HNSW subgraphs.

Algorithm 6 Bridged-HNSW Insertion [Author's research]

function INSERT(bridged_graph, q)
    $C \leftarrow compatible\_canopies(q)$
    for all $c \in C$ **do**
        $insert\_hnsw(bridged\_graph[c], q)$
    if $|C| > 1$ **then**
        $update\_bridges(q, C)$
    $update\_global\_seeds(bridged\_graph)$

Algorithm 7 Bridged-HNSW Neighborhood [Author's research]

function BRIDGED_NEIGHBORHOOD(canopies, q, layer)
    $N \leftarrow \{\}$
    for all $c \in canopies$ **do**
        $N \leftarrow N \cup \{neighborhood(bridged\_graph[c], q, layer)\}$
    return N

Algorithm 8 Bridged-HNSW Search Layer [Author's research]

function BRIDGED_SEARCH_LAYER(bridges, q, ep, ef, layer)
    $V \leftarrow \{ep\}$
    $C \leftarrow \{ep\}$
    $W \leftarrow \{ep\}$
    while $|C| > 0$ **do**
        $c \leftarrow extract\_closest(C, q)$
        $f \leftarrow get\_furthest(W, q)$
        if $distance(c, q) > distance(f, q)$ **then**
            break
        $canopies \leftarrow bridges[c]$
        $N \leftarrow bridged\_neighborhood(canopies, c, layer)$
        for all $n \in N$ **do**
            if $n \notin V$ **then**
                $V \leftarrow V \cup \{n\}$
                $f \leftarrow get\_furthest(W, q)$
                if $distance(n, q) < distance(f, q)$ **or** $|W| < ef$ **then**
                    $C \leftarrow C \cup \{n\}$
                    $W \leftarrow W \cup \{n\}$
                    if $|W| > ef$ **then**
                        $remove(W, f)$
    return W

Algorithm 9 Bridged-HNSW Query [Author's research]

function BRIDGED_HNSW_SEARCH(bridged_graph, q, k, efSearch)
  $canopy, seed \leftarrow select\_closest(canopy\_seeds, q)$
  $ep \leftarrow seed$
  for $layer = L$ **down to** 1 **do**
    $ep \leftarrow search\_layer(bridged\_graph[canopy], q, ep, ef = 1, layer)$
  $W \leftarrow bridged\_search\_layer(bridges, q, ep, ef = efSearch, layer = 0)$
  return $top\_k(W, k)$

By combining localized canopy graphs with bridging and global seeding, this variant aims to retain scalability benefits while preserving high recall across canopy boundaries. It therefore represents the most sophisticated of the three approaches and provides the main focus for this comparative analysis.

4.3. Theoretical Complexity

The theoretical complexity analysis provides an abstract view of the expected computational costs and memory requirements associated with each algorithmic variant. While experimental evaluation is essential to assess practical performance, asymptotic formulas make it possible to highlight the main trade-offs introduced by canopy-based strategies in comparison to pure HNSW.

In particular, the influence of vector replication across canopies (α) and the overhead of additional bridges and global seeds (γ) on insertion, query, and memory complexity is examined. This analysis provides the foundation for interpreting the subsequent empirical results.

4.3.1. Baseline HNSW

- Build per insertion: $O(M \cdot efConstruction \cdot \log S)$
- Build total: $O(S \cdot M \cdot efConstruction \cdot \log S)$
- Query: $O(efSearch \cdot \log S)$
- Memory: $O(S \cdot M)$

4.3.2. Canopy-Based Multi-Graph HNSW

- Build per insertion: $O(\alpha \cdot M \cdot efConstruction \cdot \log(S/C))$
- Build total: $O(S \cdot \alpha \cdot M \cdot efConstruction \cdot \log(S/C))$
- Query: $O((C \cdot efSearch \cdot \log(S/C) + Ck \log(Ck)))$
- Memory: $O(\alpha \cdot S \cdot M)$

4.3.3. Bridged-HNSW

- Build per insertion: $O(\gamma + \alpha \cdot M \cdot efConstruction \cdot \log(S/C))$
- Build total: $O(S \cdot (\gamma + \alpha \cdot M \cdot efConstruction \cdot \log(S/C)))$

- Query: $O(efSearch \cdot \log(\gamma \cdot S/C))$
- Memory: $O(\alpha \cdot S \cdot M + \gamma \cdot S)$

In summary, Baseline HNSW exhibits logarithmic query complexity and linearithmic build cost but suffers from high construction overhead at large scales. The Canopy-Based Multi-Graph HNSW reduces the cost of individual insertions by operating on smaller subgraphs and can benefit from parallelism, though replication of vectors (α) and the need to merge partial results introduce additional costs. The Bridged-HNSW balances these aspects by combining canopy-level scalability with global connectivity through bridges and seeds, which improves recall at the expense of extra memory overhead (γ) and potential exploration of multiple canopies during queries. These distinctions elucidate the theoretical trade-offs that are empirically investigated in this study.

4.4. Expected Trade-offs

The three algorithms exhibit distinct trade-offs in accuracy, scalability, and computational performance. The Pure HNSW variant is typically the fastest option for small and medium-scale datasets, delivering high efficiency in both index construction and query execution. However, its performance may degrade in terms of recall as datasets grow very large, due to the increasing cost of maintaining and navigating a single monolithic structure.

The Canopy-Based Multi-Graph HNSW, in turn, emphasizes scalability by decomposing the dataset into smaller subgraphs and exploiting parallelism during both construction and search. This approach reduces the computational burden per subgraph, but it introduces the need to merge the partial results obtained from each canopy. Consequently, query time may be negatively affected when a large number of canopies exist.

Finally, the Bridged-HNSW provides an intermediate solution between these two extremes. By enabling cross-canopy traversal through shared elements and global seeds, it preserves scalability while sustaining high recall levels. This additional connectivity comes at the expense of higher memory consumption and slightly increased query times, but it may provide a more balanced solution for large-scale, high-dimensional search tasks.

4.5. Expected results

This research aims to advance both the algorithmic foundations of Approximate Nearest Neighbor Search and its practical application in Information Systems by proposing a novel integration of Canopy Clustering with graph-based ANNS methods. The contribution is twofold: first, it may demonstrate that overlapping clustering can serve as an effective divide-and-conquer heuristic to lower the computational burden of graph construction; second, it may show that this reduction can be achieved while maintaining accuracy levels that remain acceptable for IS applications. By repositioning Canopy Clustering from its traditional role in preprocessing for K-means to the domain of nearest neighbor indexing, the study intends to foster the existing body of work on scalability in high-dimensional search.

4.6. Practical Aspects of the Implementation

In addition to the algorithmic design, practical decisions were necessary to implement the proposed canopy-guided HNSW variants. These details concern how canopy thresholds

were defined and, in the Bridged-HNSW implementation, how global connectivity was enforced. Furthermore, the implementation incorporates optimizations aimed at scalability in all produced experiments. The main aspects are summarized below.

Thresholds T_1 and T_2. The thresholds needed to construct canopies were not defined arbitrarily, but adaptively calculated from the dimensionality of the vector space. Based on the statistical distribution of angles between random vectors in high dimensions, the algorithm estimates percentiles of the angular distribution and converts them into cosine distances. In practice, T_1 is defined by a chosen percentile (e.g., 10%), while T_2 is set to half that percentile, ensuring controlled overlap between canopies. This adaptive approach avoids the pitfalls of fixed thresholds and naturally adapts to different dimensionalities.

Seeds and bridges. Every canopy is anchored by a seed element, consistently inserted across all graph layers. These seeds serve as stable entry points for search and are cached to accelerate distance computations. Furthermore, when a vector belongs to multiple canopies, bridge connections are automatically created between their respective subgraphs. These shared seeds and bridges among canopies implement the global connectivity required in the Bridged-HNSW variant, enabling queries to traverse between regions of the index.

Optimizations. Optimization measures were adopted to mitigate Python's inherent overhead, such as the use of `Numba`, memory-efficient class definitions, and heap data structures for candidate management. These optimizations make the implementation suitable for the purposes of experimental validation.

4.7. Experimental Environment

All experiments were implemented in Python 3.8.7, making use of the `NumPy` library (version 1.22.4) for vector operations and `Numba` (version 0.58.1) for just-in-time compilation to accelerate distance computations. Development and execution were carried out in the PyCharm IDE (version 2020.3.3 Community Edition) on a workstation running Windows 10 (64-bit), equipped with an Intel Core i7-10510U CPU (1.80–2.30GHz) and 20 GB of RAM.

It is acknowledged that Python is not the most suitable language for achieving maximum computational performance in large-scale graph construction and search. Nevertheless, all experiments were carried out under the same implementation and hardware conditions, ensuring fairness in the comparative analysis.

5. Results

The comparative analysis of the three algorithmic variants reveals distinct trade-offs between construction efficiency, recall, and query latency. From an IS perspective, these trade-offs are not merely technical: they also translate into organizational costs (infrastructure usage, update cycles) and user experience (responsiveness and trust in retrieved results).

Regarding indexing time (Figure 1), the Baseline HNSW consistently achieved the lowest build times across all dataset sizes, while the Bridged-HNSW presented the highest costs. The Multi-Graph HNSW exhibited intermediate performance, with build times situated between the other two variants. This result highlights the efficiency of maintaining a single monolithic structure in the baseline approach, though at the expense of recall.

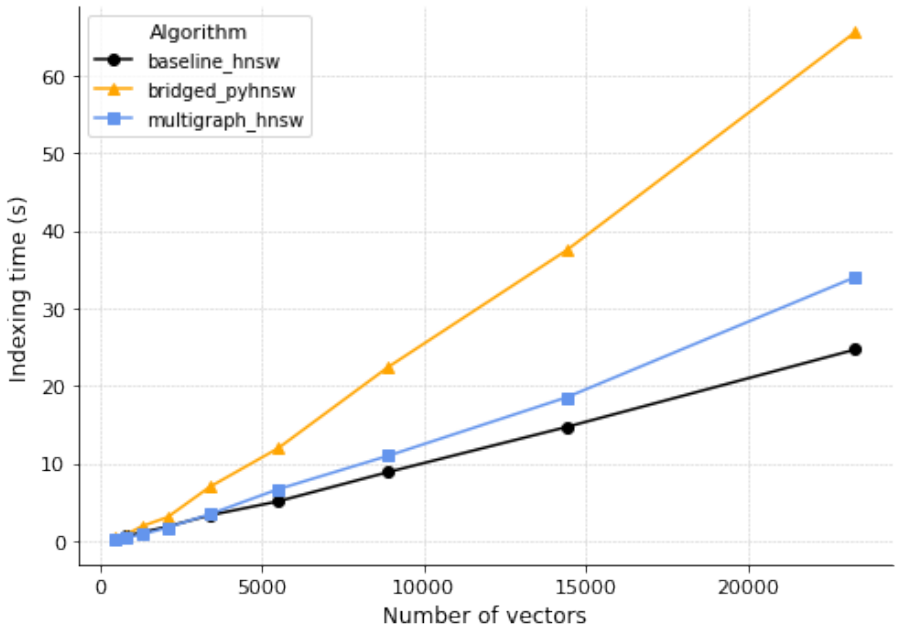

Figure 1. Lowest indexing time per algorithm and number of vectors

In terms of recall@100 (Figure 2), the Multi-Graph HNSW obtained the best results overall, particularly for datasets larger than 1,000 vectors. While the Baseline HNSW achieved the lowest recall rates, the Bridged-HNSW offered intermediate performance, benefiting from its bridging mechanism to mitigate losses. Notably, the Multi-Graph HNSW reached a recall of 99% for datasets of 1,300, 2,100, and 3,400 vectors. Although recall steadily decreased with dataset growth — dropping to 19% recall@100 for 23,300 vectors, meaning that only 19 of the 100 exact nearest neighbors were retrieved — it still outperformed the other variants under these conditions. At that scale, the Baseline HNSW recovered only 1 exact neighbor, while the Bridged-HNSW retrieved 8.

When considering indexing time at the highest observed recall (Figure 3), all three algorithms exhibited the expected growth with dataset size. However, Bridged-HNSW displayed the steepest increase, followed by the Baseline HNSW. The Multi-Graph HNSW, in contrast, exhibited comparatively more controlled growth, remaining more efficient at large scales in terms of balancing recall and construction costs. From an organizational perspective, this balance is relevant: sustaining high recall with lower construction overhead makes frequent index refresh cycles more affordable, which can improve the freshness of information available to users without imposing prohibitive infrastructure costs.

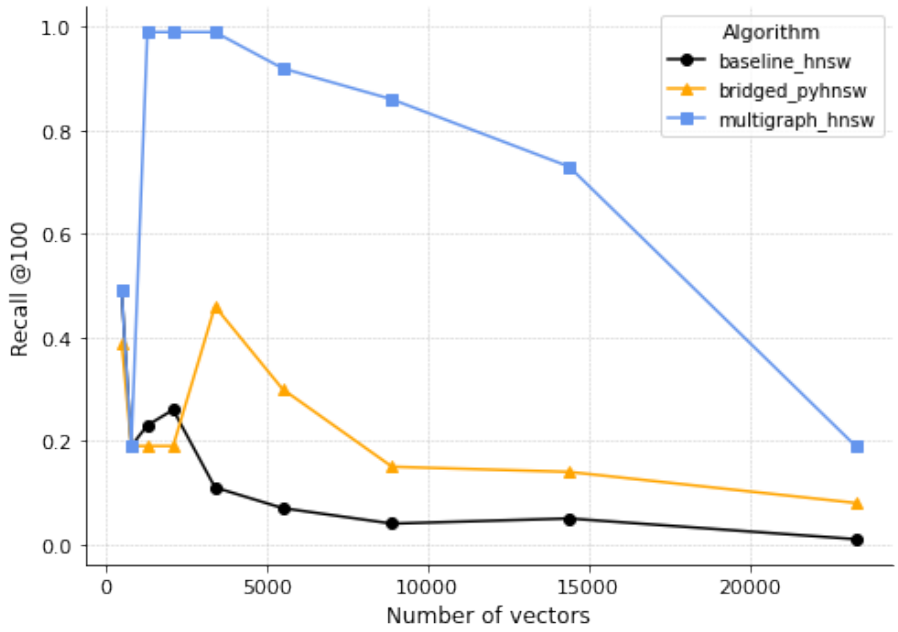

Figure 2. Recall at the lowest indexing time per algorithm and number of vectors

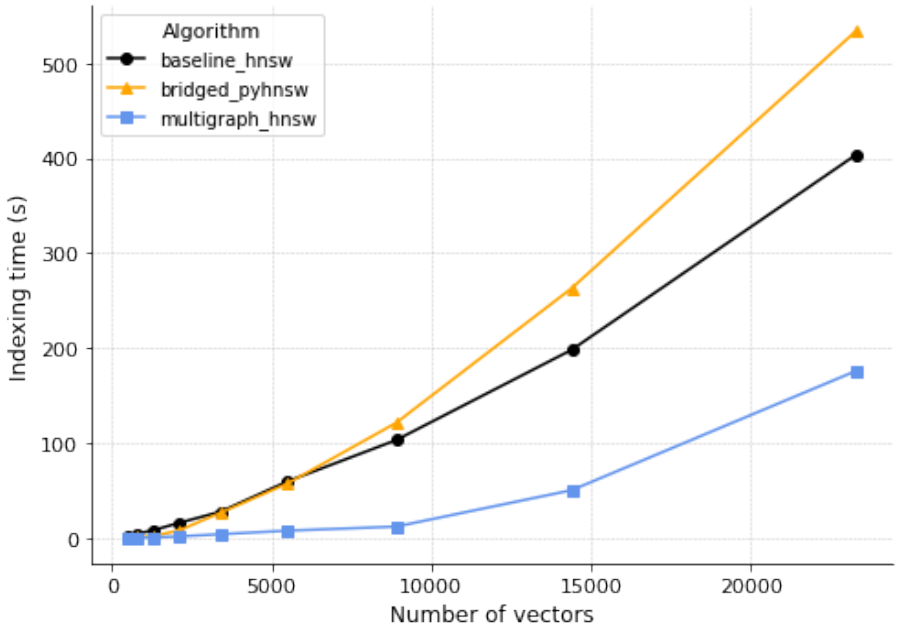

Figure 3. Indexing time at the highest recall per algorithm and number of vectors

In contrast, when analyzing the lowest indexing time under minimum acceptable recall thresholds (Figure 4), the Baseline HNSW exhibits the highest construction times, while the Multi-Graph still achieves the lowest costs.

A different picture emerges when examining the respective query times of each algorithm under those same conditions (Figure 5). In this scenario, Baseline HNSW presented the most stable growth in mean query time as dataset size increased, even though its recall degraded significantly on larger collections. The Bridged-HNSW demonstrated

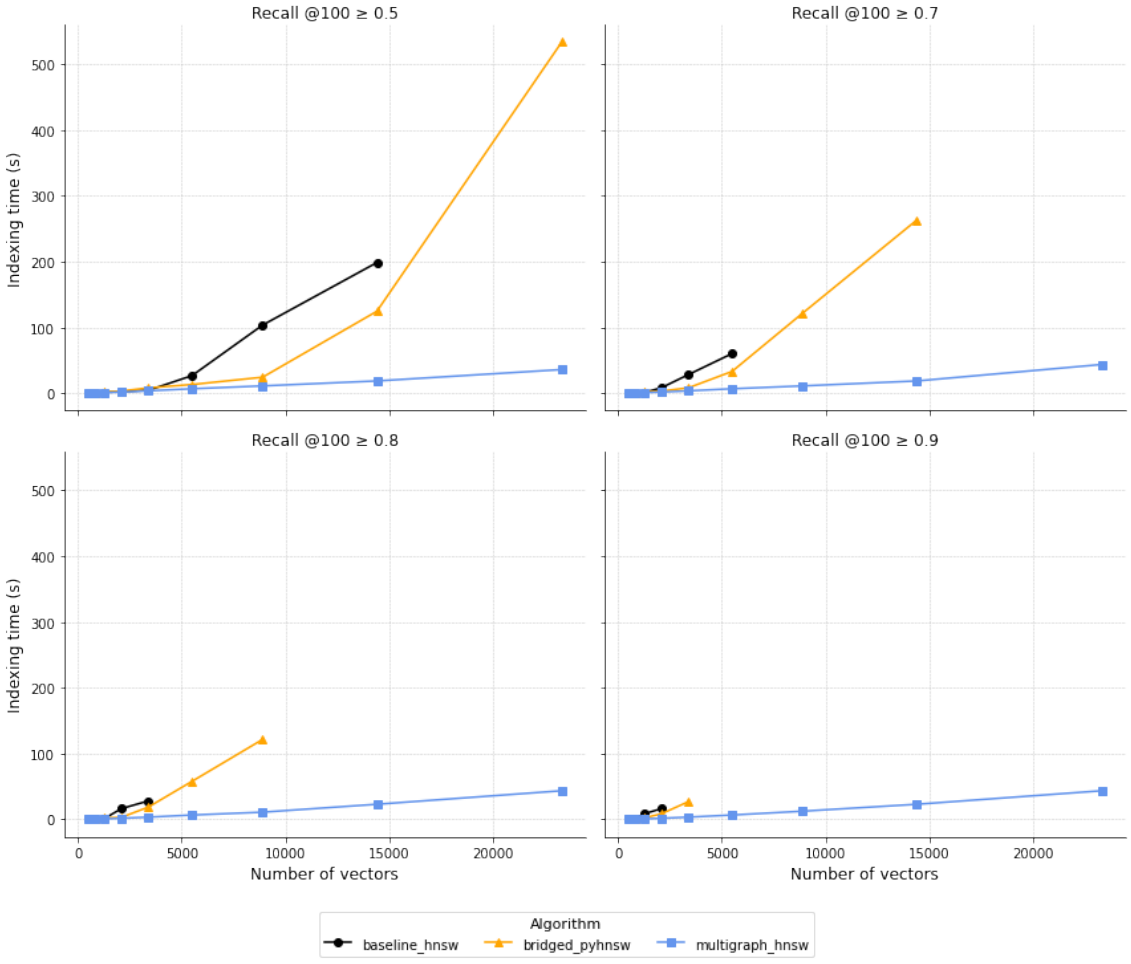

Figure 4. Lowest indexing time under minimum acceptable recall thresholds

behavior similar to the baseline but with the advantage of sustaining higher recall, especially on datasets exceeding 2,000 vectors. Conversely, the Multi-Graph HNSW exhibited sharply increasing query times as dataset size grew, indicating search scalability limitations despite strong recall performance.

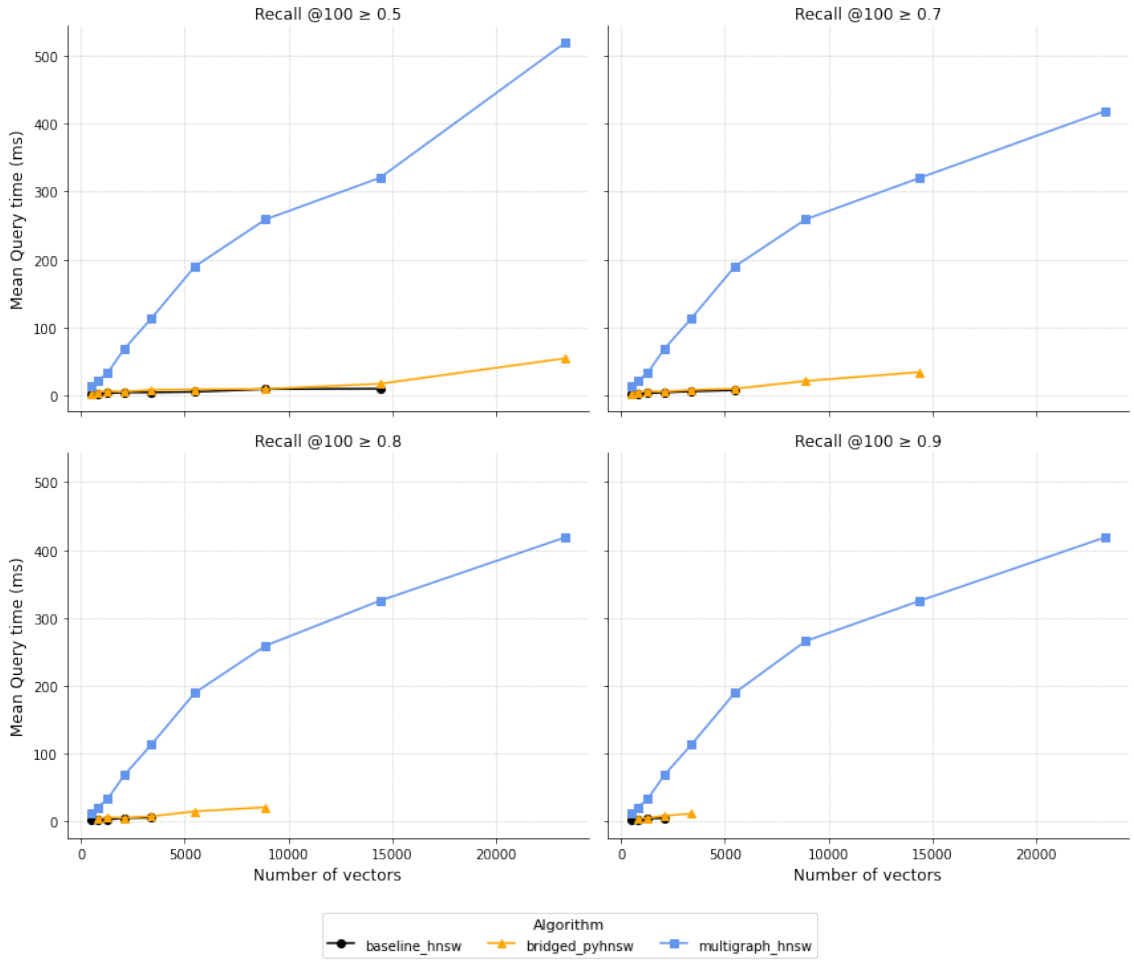

Figure 5. Query time at the lowest indexing time under minimum acceptable recall thresholds

These trends are particularly evident in the three largest datasets via dispersion plots correlating recall@100 with indexing time and also query latency with indexing time (Figure 6). Together, the results suggest that the Multi-Graph HNSW is most suitable when recall is the primary concern, while the Baseline HNSW remains advantageous when minimizing query latency is critical. The Bridged-HNSW provides a balanced compromise, sustaining higher recall than the baseline while avoiding the query-time overheads observed in the multi-graph approach. From an IS adoption standpoint, this reinforces that algorithm choice depends on organizational priorities: environments such as product recommendation may tolerate lower recall in favor of faster responses, whereas knowledge management or decision-support systems may demand higher recall even at the cost of increased latency.

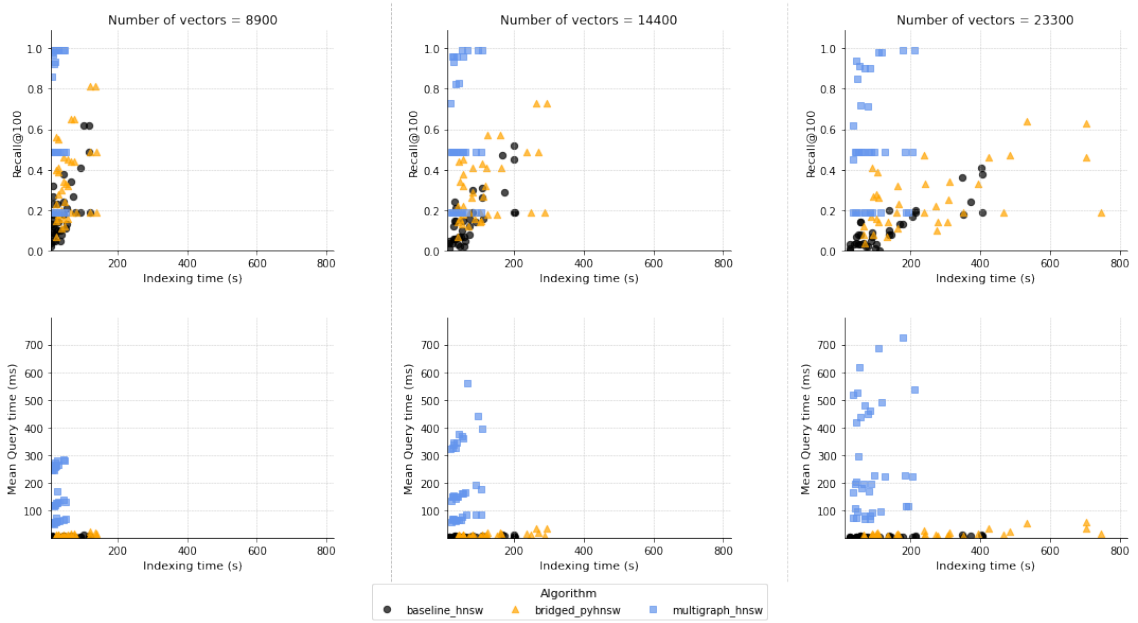

Figure 6. Dispersion of metrics for the three largest datasets

The empirical findings corroborate the trade-offs anticipated in Section 4.4 and the expected contributions outlined in Section 4.5. As predicted, the Baseline HNSW excelled in indexing and query times for small and medium-scale datasets, but its recall degraded substantially as the dataset size increased, confirming the limitations of a monolithic index structure. The Multi-Graph HNSW, aligned with expectations of enhanced scalability through decomposition, achieved superior recall by leveraging overlapping canopies; however, it incurred the cost of merging results, reflected in its steep query-time growth on larger datasets. Finally, the Bridged-HNSW confirmed its role as a balanced compromise: although its construction time was higher than the other variants, the combination of canopy-level partitioning with bridging mechanisms allowed it to sustain recall advantages over the Baseline while avoiding the extreme query-time penalties of the Multi-Graph approach.

Taken together, these observations reinforce the proposition that Canopy Clustering can indeed function as an effective divide-and-conquer heuristic, reducing construction costs while maintaining accuracy levels suitable for Information Systems applications. Beyond the technical dimension, the results illustrate sociotechnical trade-offs: lower construction costs directly reduce infrastructure demands and support more frequent updates in organizational processes, while lower query latency improves responsiveness and user satisfaction. In this sense, the study advances the IS field not only technologically but also by contributing to processes and people that respectively rely on and use intelligent and data-driven decision making systems.

Moreover, the findings align with IS theories on knowledge retrieval efficiency and the optimization of data-intensive processes in decision-support environments, particularly by addressing trade-offs between accuracy and computational feasibility in large-scale retrieval. In this sense, the study can contribute to the IS field by introducing a novel heuristic combination that enhances the scalability of similarity search on ANNS graphs,

while also providing methodological and practical insights that may be extended to cost-effective recommendation, retrieval, and other high-dimensional data processing tasks.

6. Conclusion

This study has presented a novel approach for integrating Canopy Clustering with graph-based Approximate Nearest Neighbor Search as a means of addressing the computational bottlenecks of high-dimensional similarity search. By applying canopy clustering as a divide-and-conquer heuristic, it was demonstrated that graph construction costs can be reduced while preserving retrieval accuracy at levels suitable for Information Systems applications. This reframing of Canopy Clustering, traditionally associated with clustering tasks such as K-means initialization, represents a methodological alternative in its adaptation to the ANNS domain, aiming at a scalable and lightweight solution to one of the most pressing challenges in large-scale vector search.

The experimental results confirmed these expectations. The Baseline HNSW offered the lowest build and query times but showed significant recall degradation on larger datasets. The Multi-Graph HNSW achieved the best recall across most scenarios, validating the benefits of canopy decomposition, though at the expense of substantially higher query times as dataset size increased. The Bridged-HNSW emerged as an intermediate solution, combining stable recall with query performance closer to the baseline, although at the expense of higher construction costs. Viewed in practice, these differences translate into concrete design choices for solutions: lower construction time reduces infrastructure consumption and enables more frequent index refresh cycles; higher recall supports the quality of recommendations and retrieval outputs consumed by analysts and end users; and lower query time improves perceived responsiveness in interactive applications. Together, these findings highlight the practical trade-offs of each approach and support the feasibility of canopy-guided graph construction as a means of balancing efficiency and accuracy in high-dimensional search.

Beyond these perspectives, future work can extend the evaluation to large-scale real-world datasets, enabling an assessment of scalability under production conditions. Another research approach concerns the impact of the proposed method on practical Information Systems applications, particularly in recommendation engines and Retrieval-Augmented Generation (RAG) pipelines, in which vector search is increasingly central. Finally, future work can also explore hybrid indexing strategies that combine Canopy clustering with alternative heuristics, as well as reimplementations in lower-level languages to better optimize the trade-off between accuracy, memory usage, and computational efficiency.

Supplementary Material

The supplementary material related to this paper — including the synthetic datasets, ground-truth files, and raw experimental results — is openly available through an anonymous repository to preserve double-blind review.[1]

Acknowledgments

The authors acknowledge the use of generative AI tools (ChatGPT-5, OpenAI) to assist in improving the grammar and overall clarity of the manuscript. All ideas, content, and

[1] https://osf.io/ehvzb/files/osfstorage?view_only=63468080839a48e39d4a5337a0ea8206

conclusions expressed are those of the authors.

References

Aguerrebere, C., Bhati, I. S., Hildebrand, M., Tepper, M., and Willke, T. (2023). Similarity search in the blink of an eye with compressed indices. *Proc. VLDB Endow.*, 16(11):3433–3446.

Aumüller, M., Bernhardsson, E., and Faithfull, A. (2020). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374.

Boscarioli, C., de Araujo, R. M., and Maciel, R. S. P., editors (2017). *I GranDSI-BR: Grand Research Challenges in Information Systems in Brazil 2016–2026*. Special Committee on Information Systems (CE-SI), Brazilian Computer Society (SBC), Porto Alegre, Brazil.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., 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., 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. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Fortune, S. (1995). Voronoi diagrams and delaunay triangulations. In Du, D.-Z. and Hwang, F., editors, *Computing in Euclidean Geometry*, pages 225–265. World Scientific.

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. (2020). 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. Curran Associates Inc.

Malkov, Y., Ponomarenko, A., Logvinov, A., and Krylov, V. (2014). Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, page 169–178, New York, NY, USA. Association for Computing Machinery.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Pro-*

cessing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268.

Wang, M., Xu, X., Yue, Q., and Wang, Y. (2021). A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proc. VLDB Endow.*, 14(11):1964–1978.