Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art

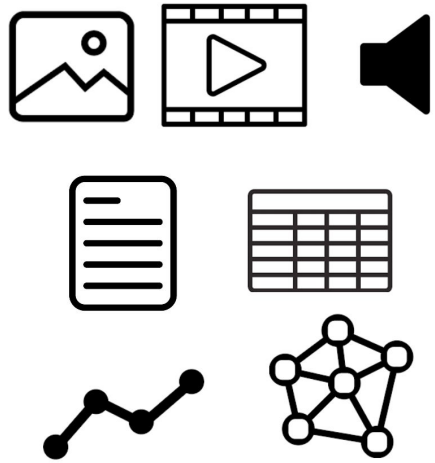
Ilias Azizi[1], Karima Echihabi[2], Themis Palpanas[3]

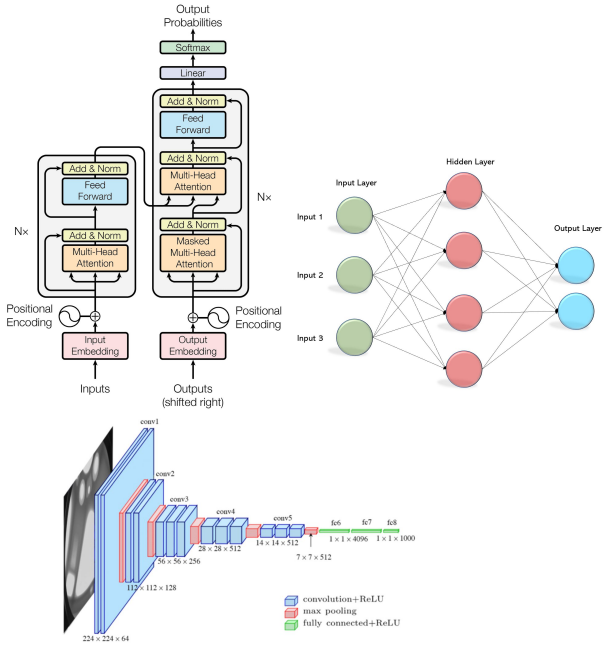
1. ETIS, ENSEA, CNRS, France
2. College of Computing, UM6P, Morocco
3. Lipade, Paris Cité University, France

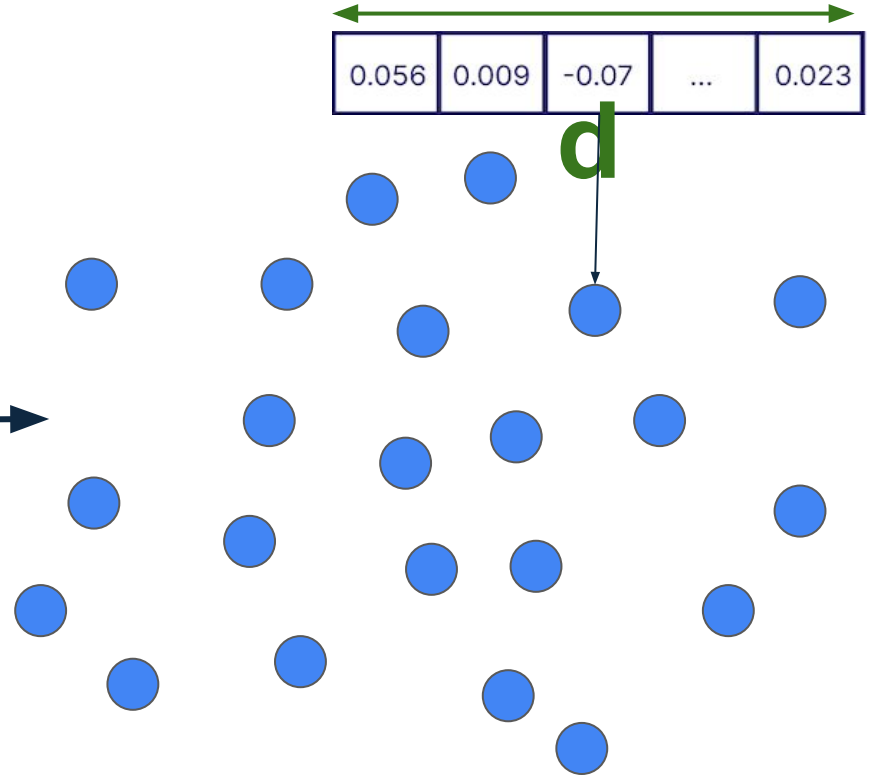
24 June 2025

Domains

Embeddings







Forecasting

Classification

Pattern recognition

Semantic Retrieval

RAG

Object Detection

RecSys

Q&A

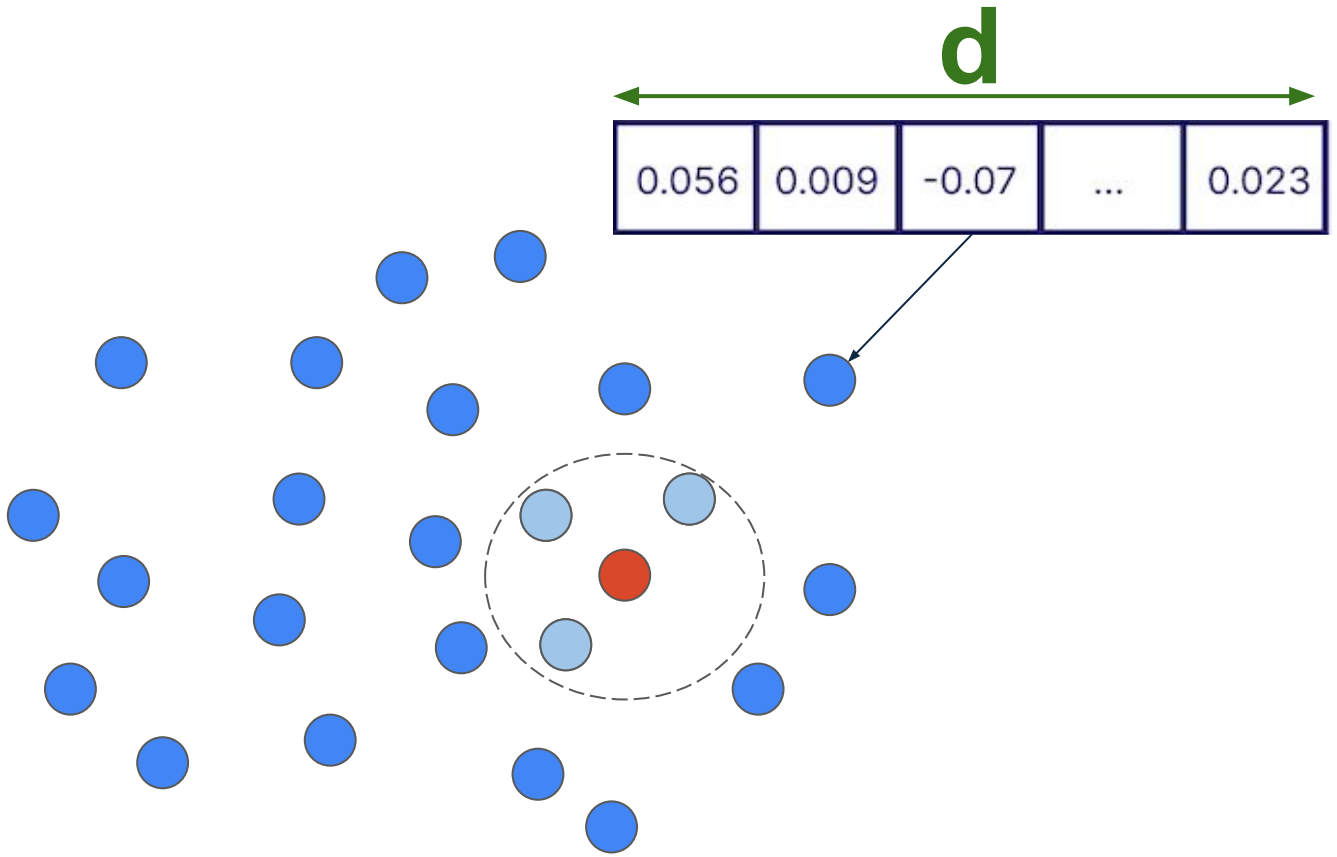
Segmentation

Generation

. . .

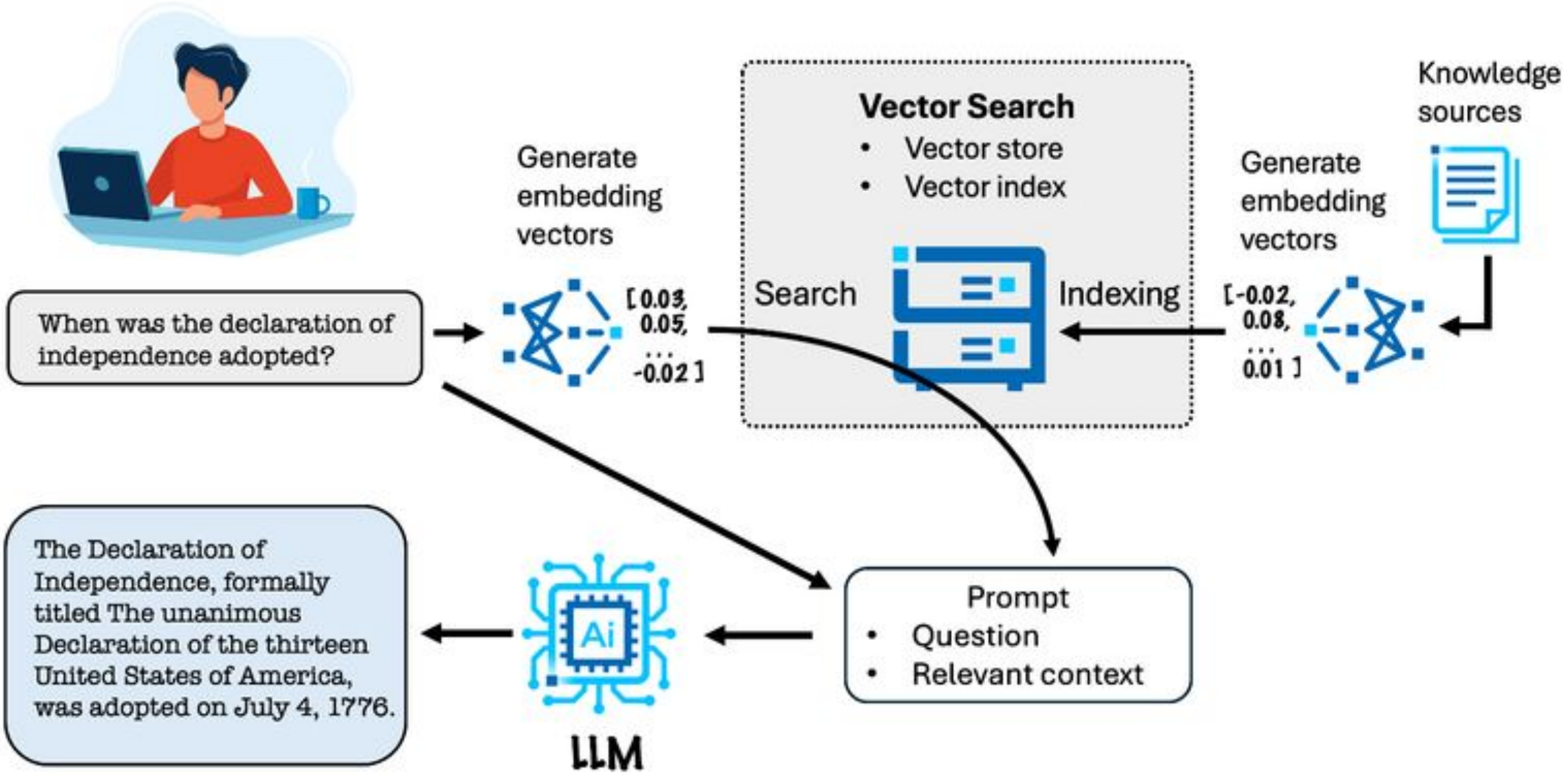
Problem

- **Given:** a set **S** of **n** distinct points in **d**-dimensional space R^d under some norm $\| . \|$
- **Goal:** return the set of K points in **S** that are **closest** to a query **Q** $\in R^d$,under $\| . \|$



$$O(n \cdot d)$$

Motivation: Retrieval-Augmented Generation



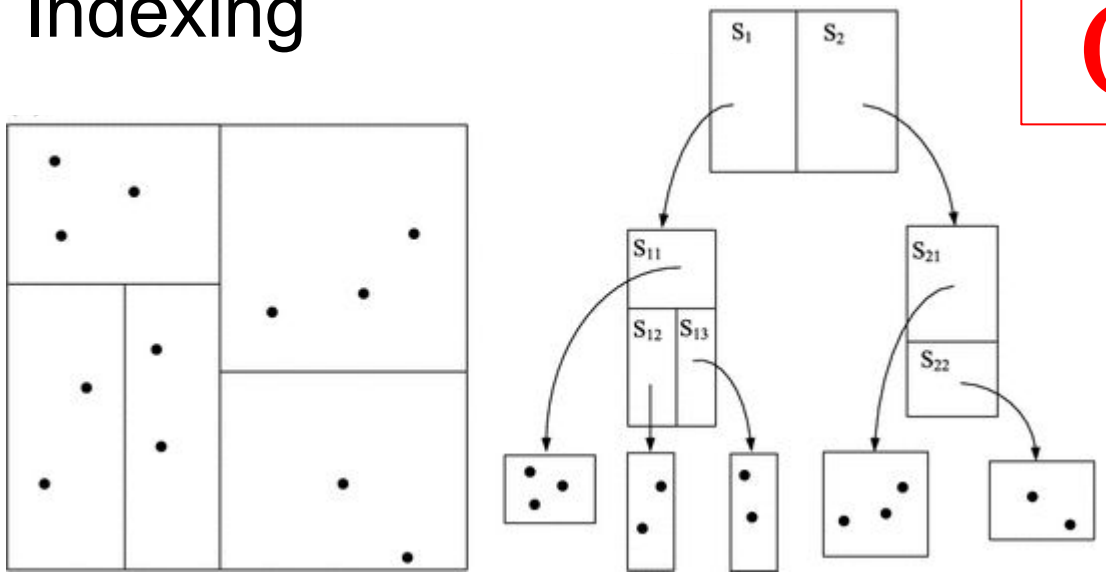
Can we do better?

Pruning non promising data points

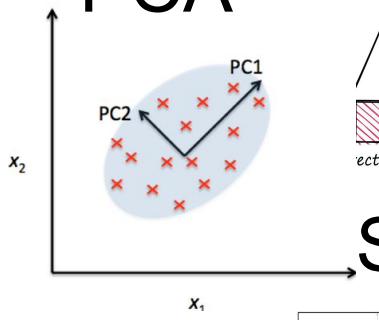
Dimension reduction

$O(n \cdot d)$

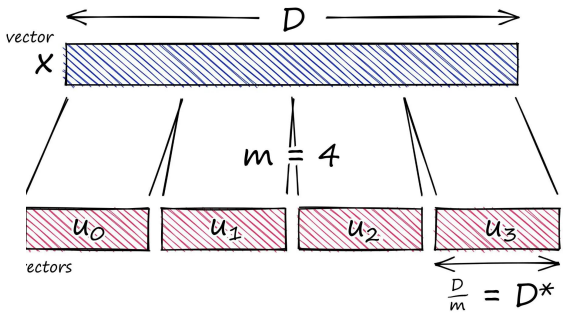
Indexing



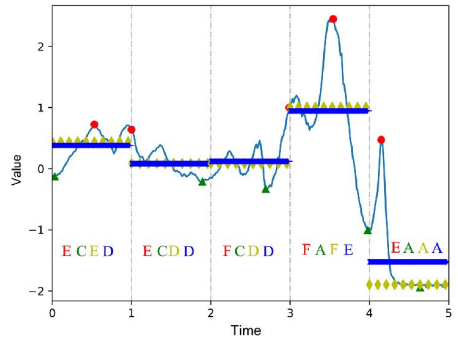
PCA



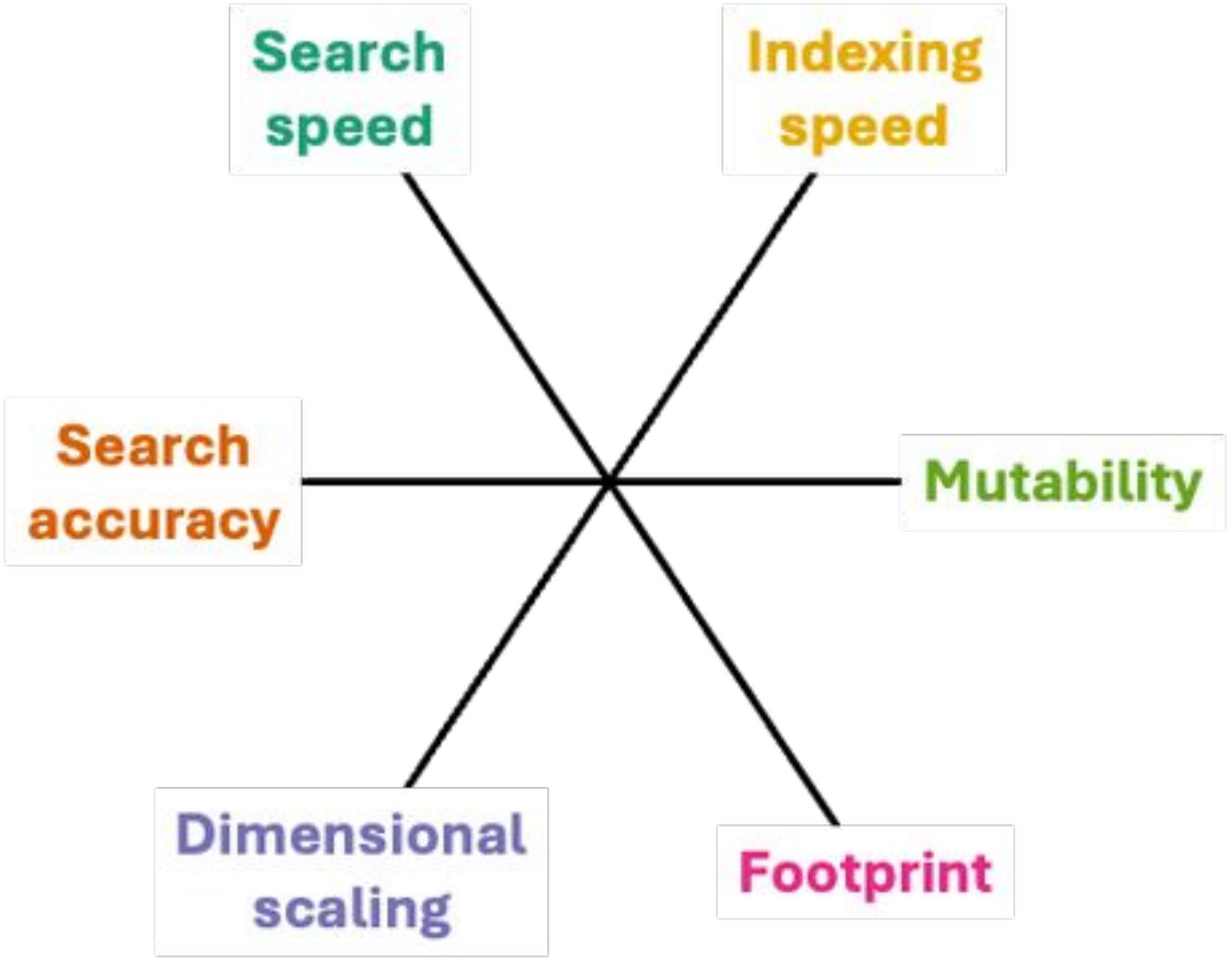
Quantization



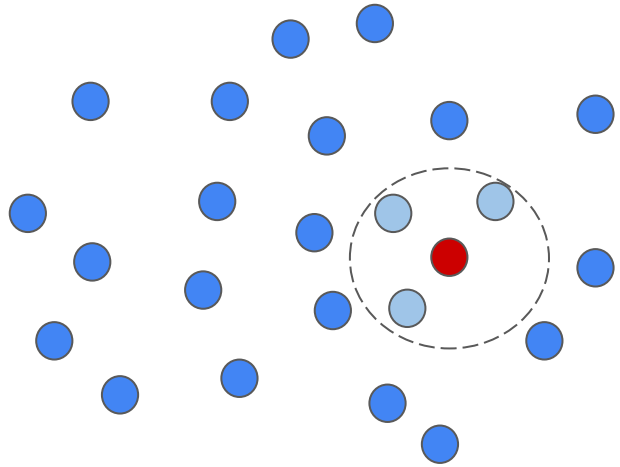
Summarization



Vector Search Approaches



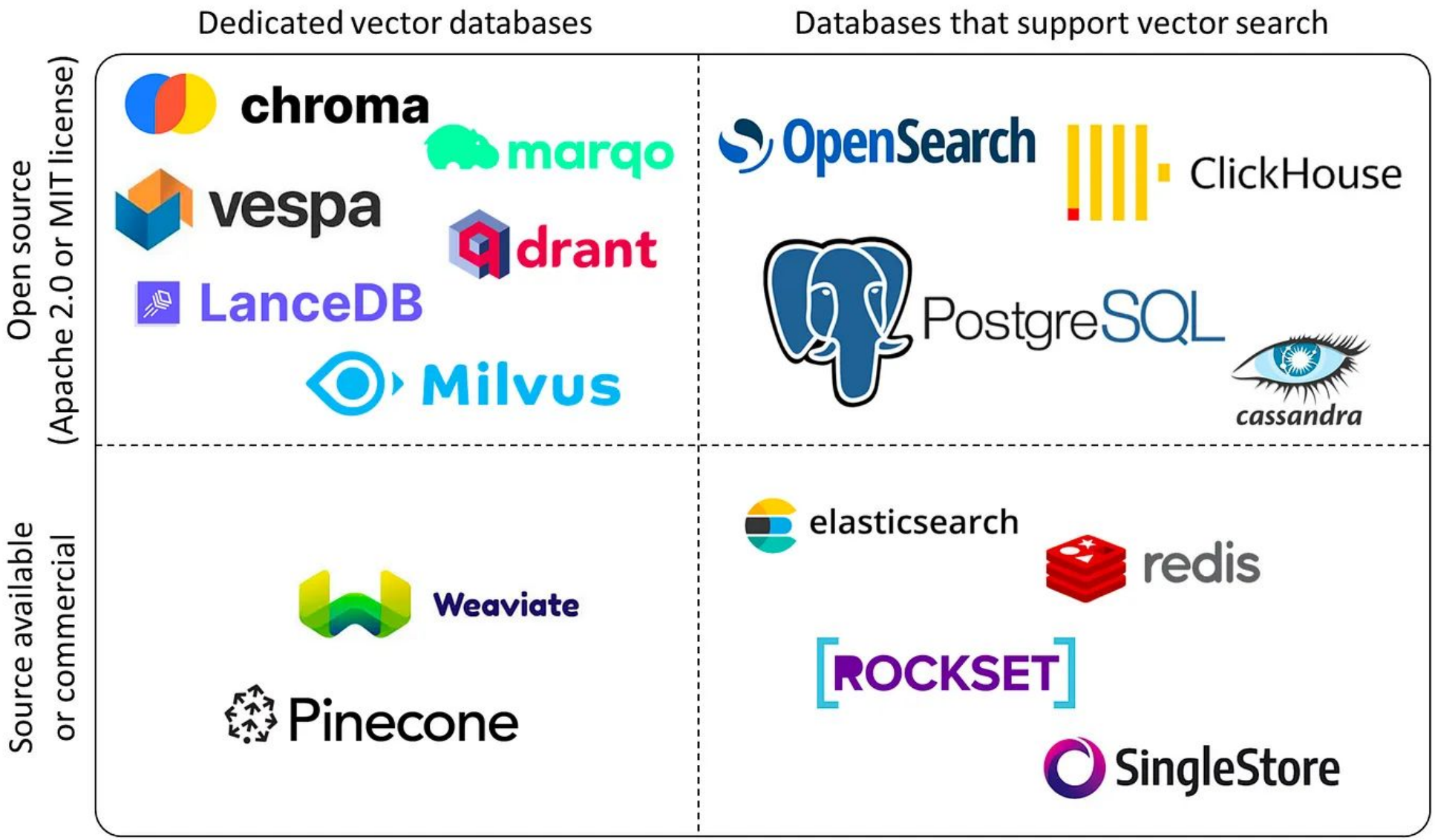
Vector Search Approaches



Vector Search Approaches

ANN Family	Advantage	Disadvantage
Tree-Based	Low index construction time Can support different flavors of search	Low search performance on hard query workloads
Hash-Based	Support theoretical guarantees on query efficiency and accuracy	High index construction time High footprint Low search performance (efficiency, accuracy)
Graph-Based	Excellent empirical query accuracy and efficiency	High Index construction time High footprint No guarantees on search quality
Inverted Index	Fast query response for sparse data	High Index construction time Less efficient for high-dimension Low search accuracy

Vector Database



Vector Database

Pinecone	Proprietary composite index
milvus / zilliz	Flat, Annoy, IVF, HNSW/RHNSW (Flat/PQ), DiskANN
Weaviate	Customized HNSW, HNSW (PQ), DiskANN (in progress...)
qdrant	Customized HNSW
chroma	HNSW
LanceDB	IVF (PQ), DiskANN (in progress...)
vespa	HNSW + BM25 hybrid
Vald	NGT
elasticsearch	Flat (brute force), HNSW
redis	Flat (brute force), HNSW
pgvector	IVF (Flat), IVF (PQ) in progress...

Image Search

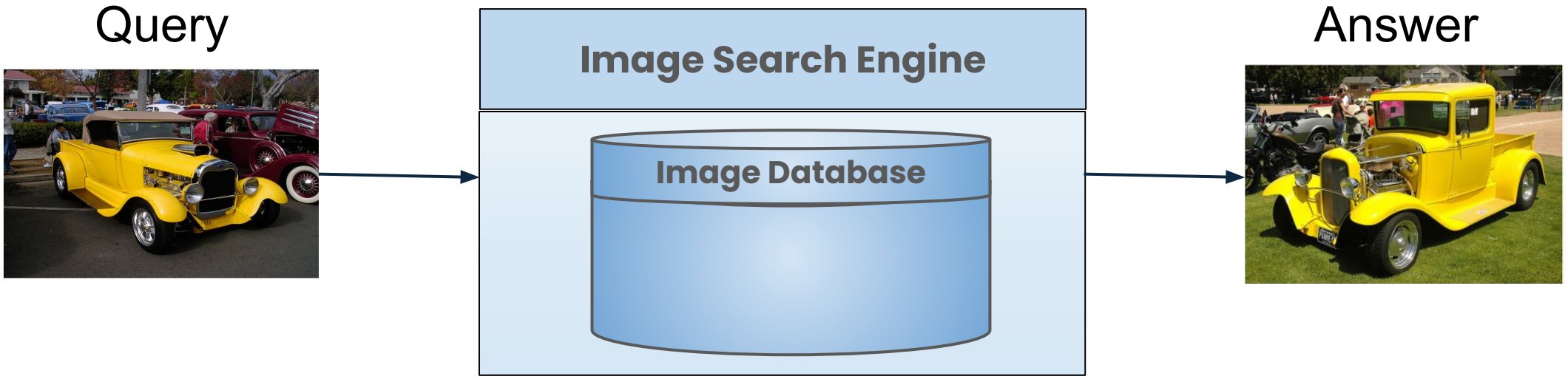
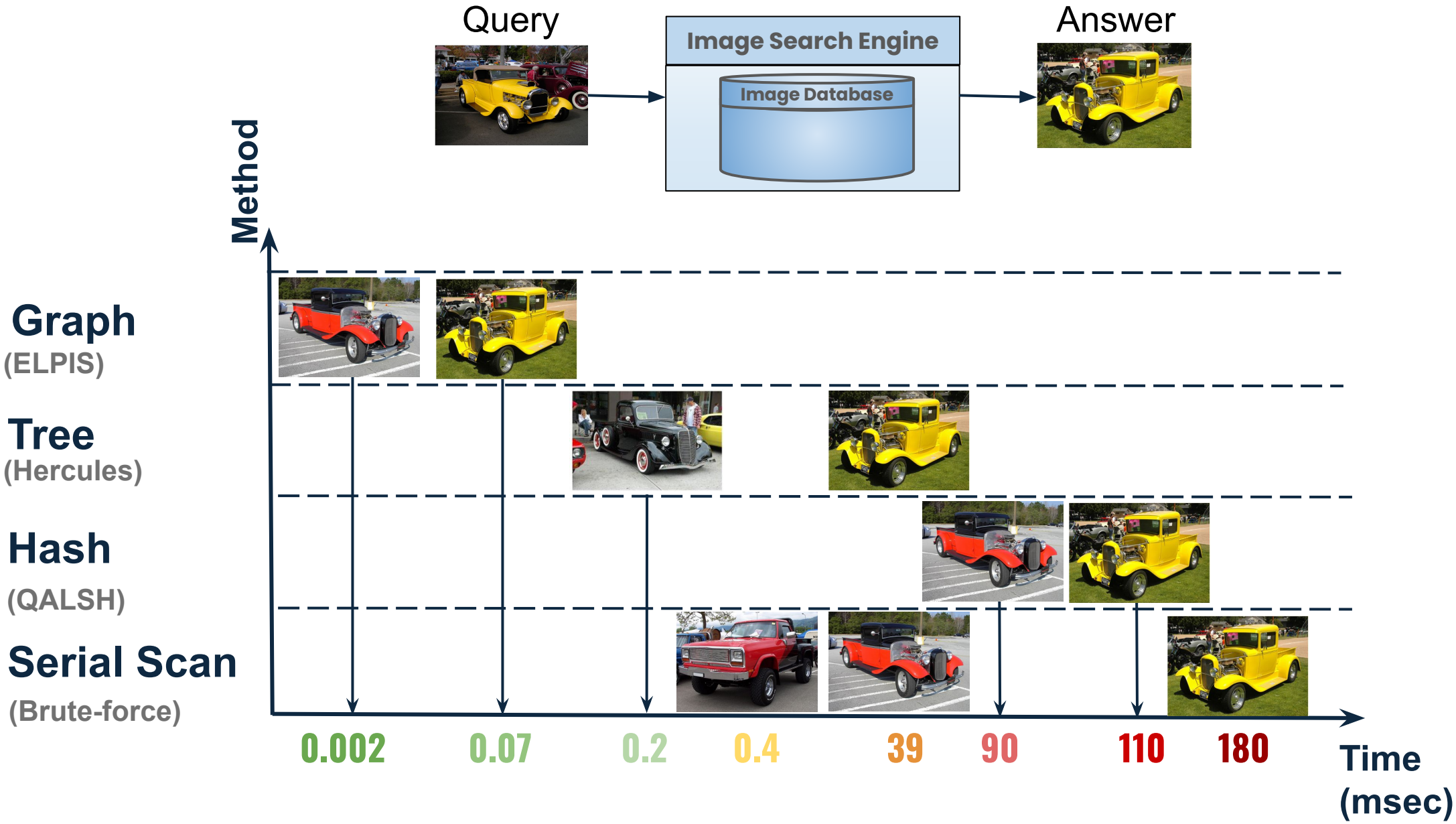


Image Search



Limitations of Existing Works

- Lack of comprehensive taxonomy
- Limited insight into graph construction’s impact on search performance.
- Key strategies underexplored
- Limited dataset diversity

Contributions

- New Taxonomy of five design paradigms
- New Insights on Key design choices
- Exhaustive experimental evaluation
- Practical guidance and recommendations

Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art

ILIAS AZIZI, UM6P, Université Paris Cité, Morocco - France
KARIMA ECHIHABI, UM6P, Morocco
THEMIS PALPANAS, Université Paris Cité, France

Vector data is prevalent across business and scientific applications, and its popularity is growing with the proliferation of learned embeddings. Vector data collections often reach billions of vectors with thousands of dimensions, thus, increasing the complexity of their analysis. Vector search is the backbone of many critical analytical tasks, and graph-based methods have become the best choice for analytical tasks that do not require guarantees on the quality of the answers. We briefly survey in-memory graph-based vector search, outline the chronology of the different methods and classify them according to five main design paradigms: seed selection, incremental insertion, neighborhood propagation, neighborhood diversification, and divide-and-conquer. We conduct an exhaustive experimental evaluation of twelve state-of-the-art methods on seven real data collections, with sizes up to 1 billion vectors. We share key insights about the strengths and limitations of these methods; e.g., the best approaches are typically based on incremental insertion and neighborhood diversification, and the choice of the base graph can hurt scalability. Finally, we discuss open research directions, such as the importance of devising more sophisticated data-adaptive seed selection and diversification strategies.

CCS Concepts: • **Theory of computation** → **Graph algorithms analysis**; **Nearest neighbor algorithms**; **Data structures design and analysis**; • **Information systems** → **Information retrieval**; • **General and reference** → *Evaluation*; *Experimentation*.

Additional Key Words and Phrases: Vector similarity search, Approximate nearest neighbor, KNN graph analysis, Seed selection, Neighborhood diversification, Graph algorithms

ACM Reference Format:
Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2025. Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. *Proc. ACM Manag. Data* 3, 1 (SIGMOD), Article 43 (February 2025), 31 pages. https://doi.org/10.1145/3709693

1 Introduction

Vector data is common in various scientific and business domains, and its prevalence is expected to grow in the future with the proliferation of learned embeddings [43, 107]. The volume and dimensionality of this data, which can exceed multiple terabytes and thousands of dimensions, make its analysis very challenging [107]. A critical component of these data analysis tasks is vector search [40, 42, 43, 107]. It supports recommendation [91, 139], information retrieval [154], clustering [21, 150], classification [114] and outlier detection [19, 20, 22, 78, 118] in many fields including bioinformatics, computer vision, security, finance and medicine. In data integration, vector search

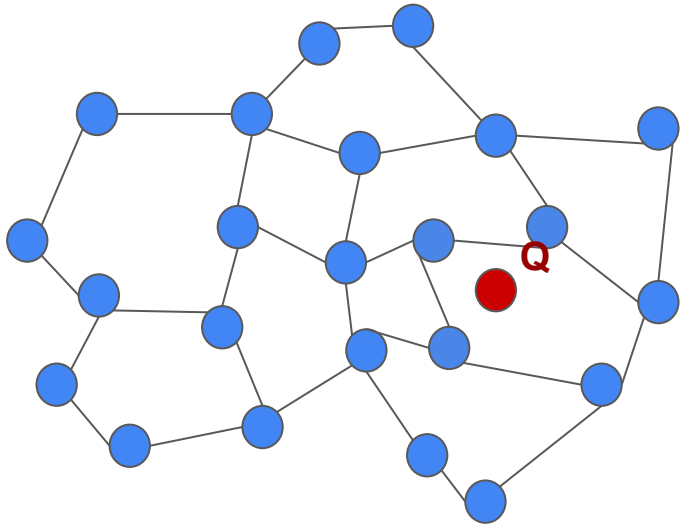
Authors' Contact Information: Ilias Azizi, UM6P, Université Paris Cité, Morocco - France, ilias.azizi@um6p.ma; Karima Echihabi, UM6P, Morocco, karima.echihabi@um6p.ma; Themis Palpanas, Université Paris Cité, France, themis@mi.parisdescartes.fr.

ACM 2836-6573/2025/2-ART43
https://doi.org/10.1145/3709693

Graph-Based Vector Search: Overview and State of the Art

Proximity Graph

Graph in which two vertices are connected by an edge if and only if the vertices satisfy particular geometric requirements



Beam Search

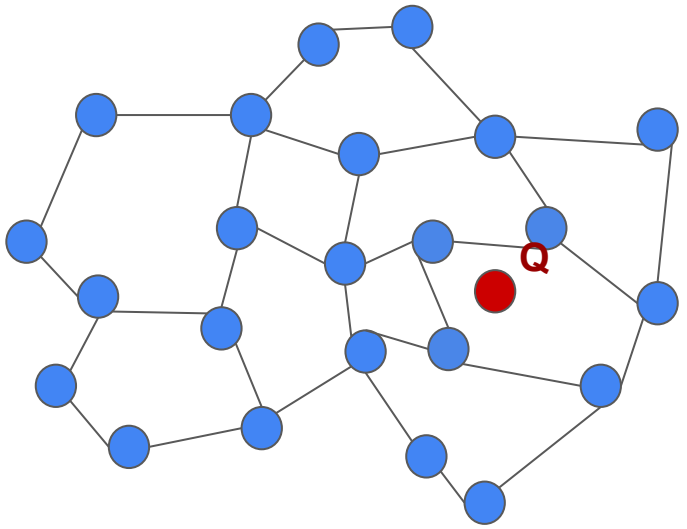
Beam search is a heuristic search algorithm that explores a graph by expanding the most optimistic node in a limited set of size L

BeamSearch(G, Q, entry_node, K, L)

Randomized Seed Selection
- K Random Sampling (KS)

Index-based Seed Selection
- Stacked NSW (SN)
- KD, KMean Balanced Trees

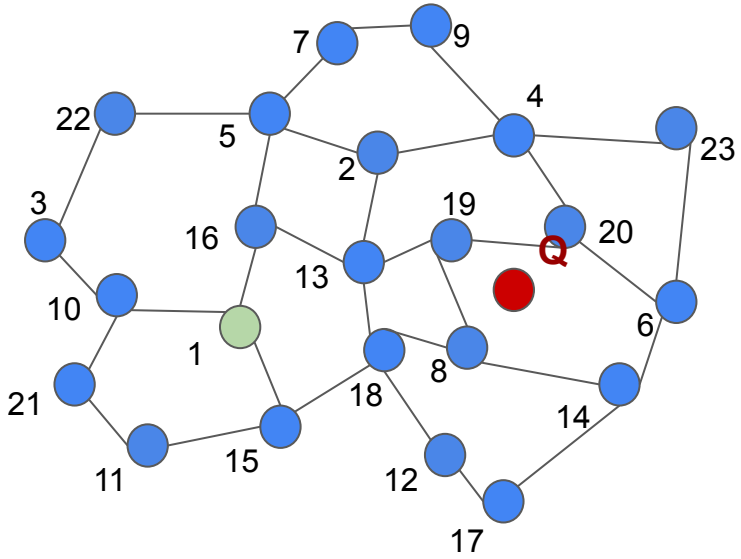
Predefined Seed Selection
- Medoid (MD)

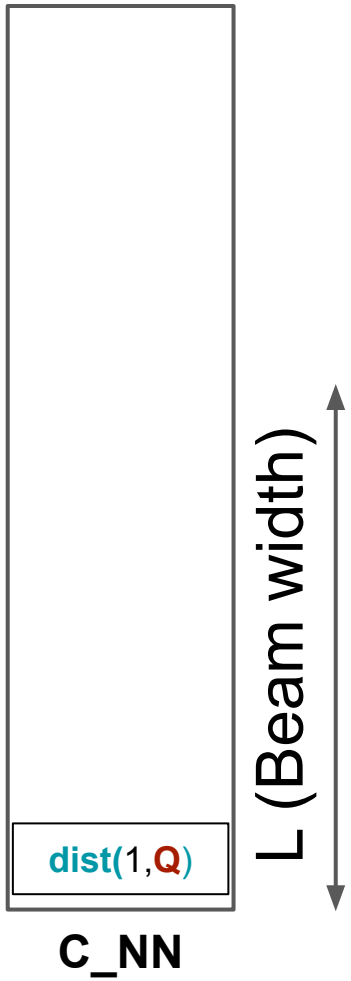


Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

1. Initialize the set of candidate nearest neighbors **C_NN** with entry_ node, and the set of visited nodes with empty set $V \leftarrow \varnothing$

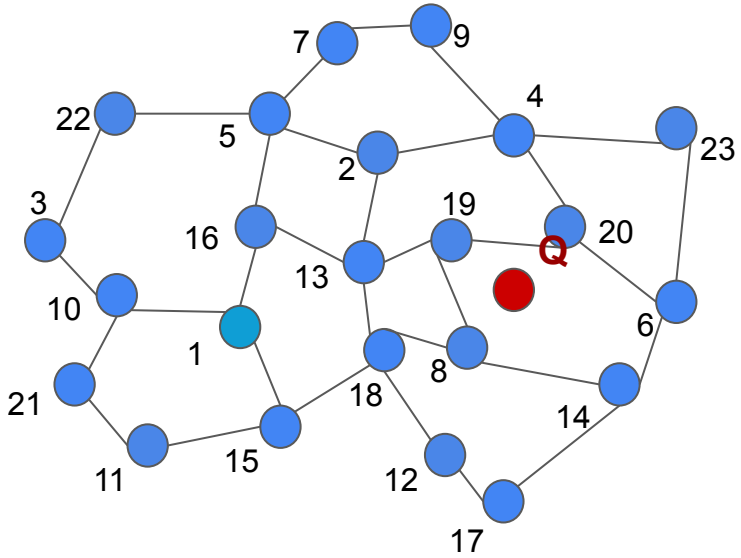




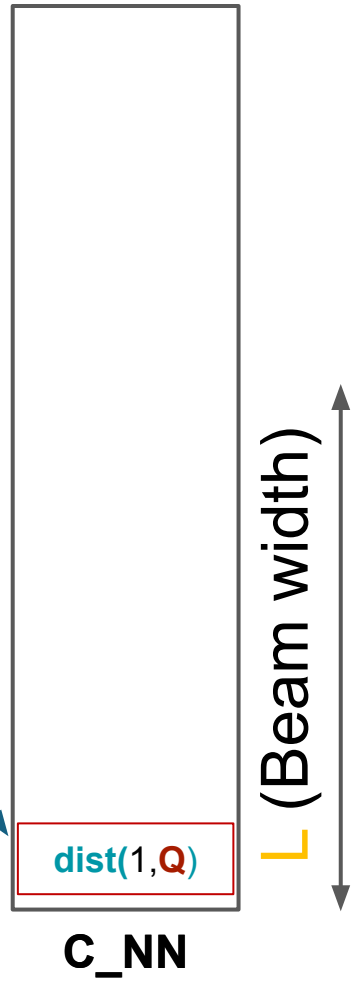
Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)



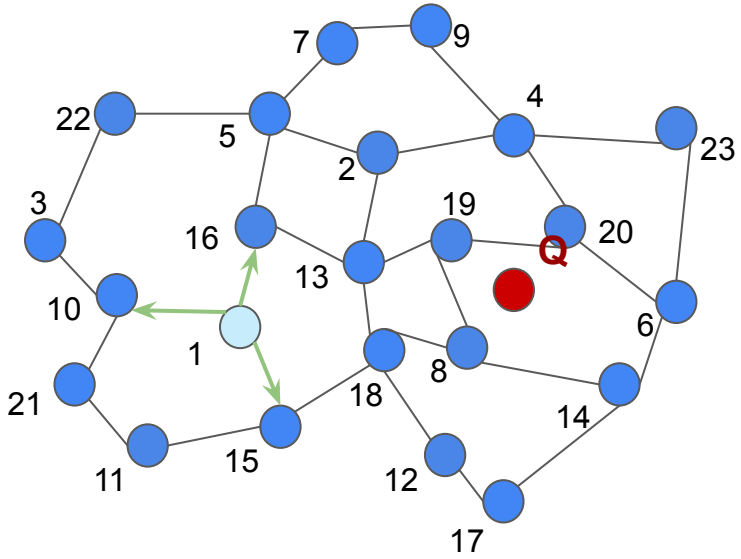
2.1 Select the closest element P from **C_NN**\V



Beam Search

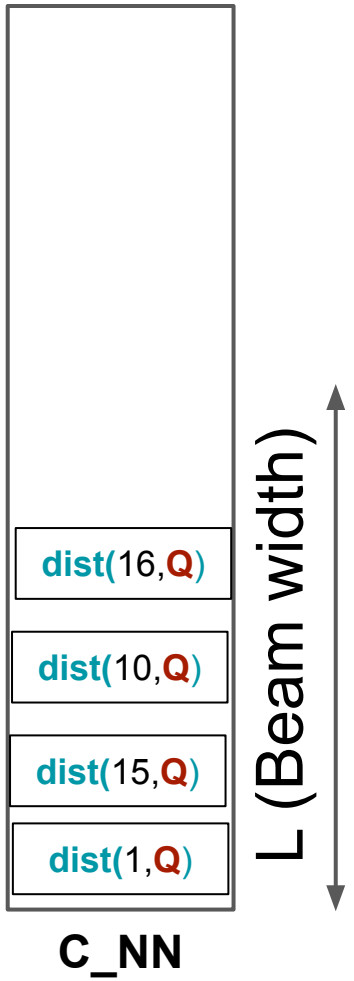
BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)



2.1 Select the closest element P from **C_NN**\V

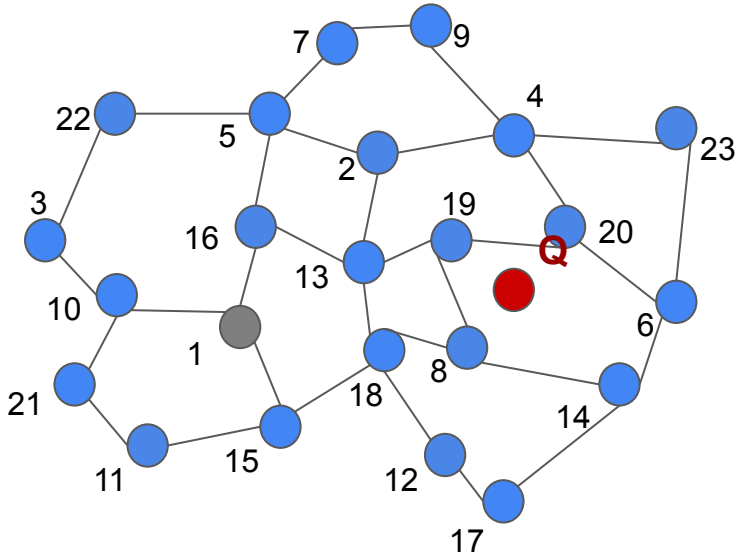
2.2 Add P neighbors to **C_NN**



Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

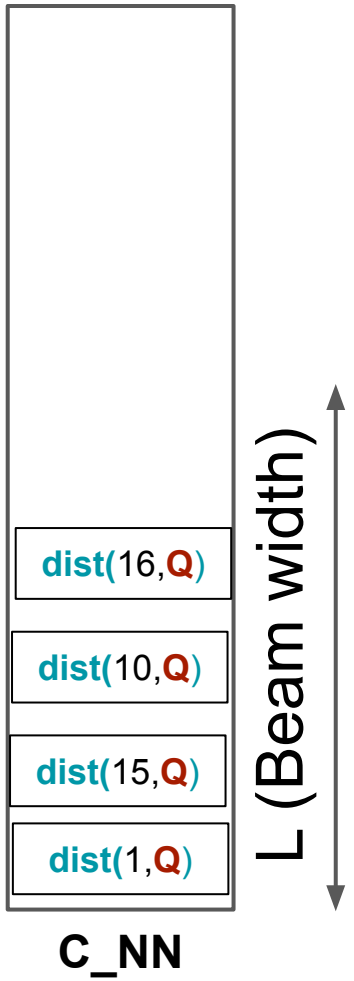
② While(**C_NN**\V≠∅)



2.1 Select the closest element P from **C_NN**\V

2.2 Add P neighbors to **C_NN**

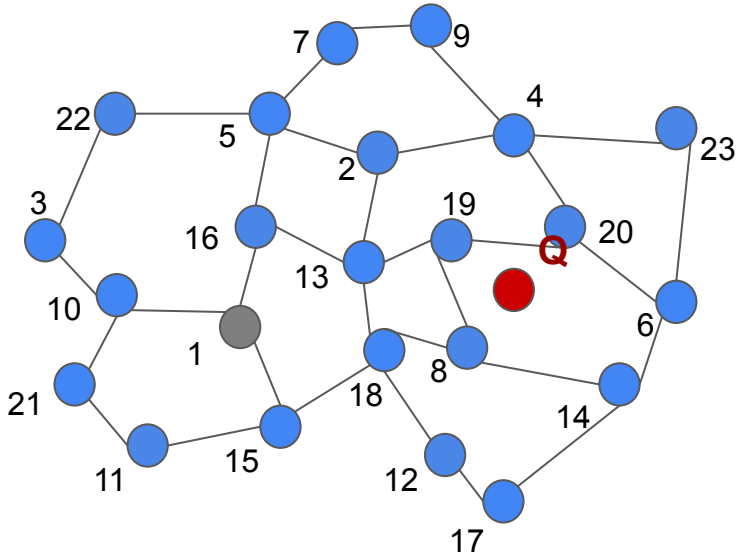
2.3 Add P to the set of visited candidate



Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)

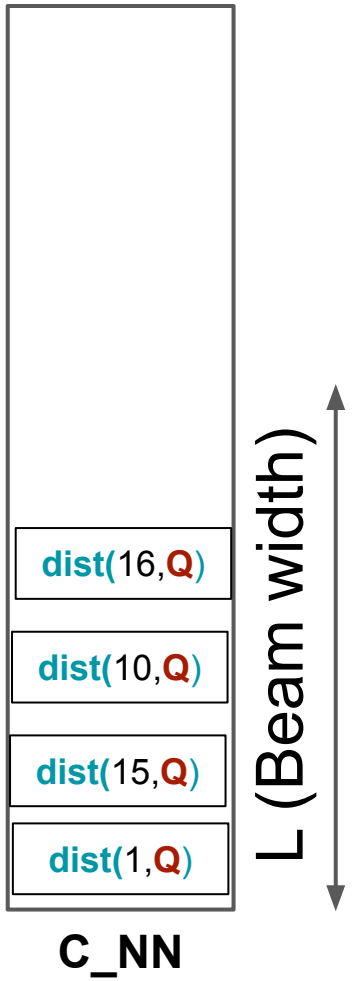


2.1 Select the closest element P from **C_NN**\V

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

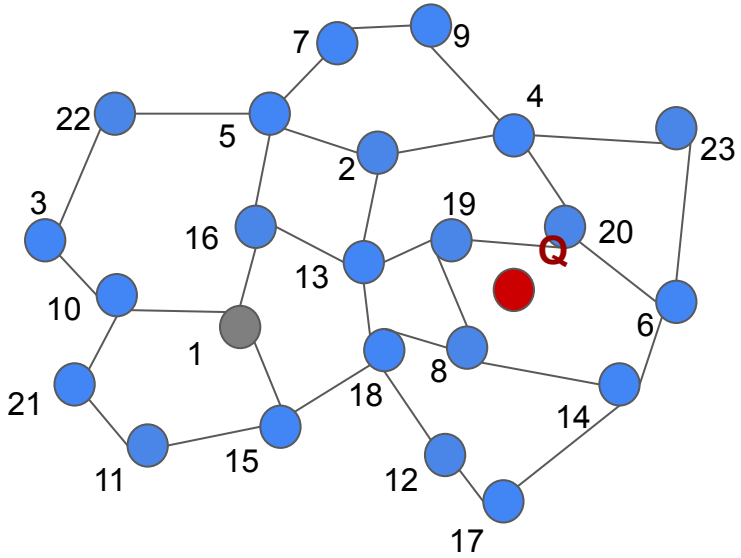
2.4 If |**C_NN**| > **L**: Update **C_NN** to retain closest **L** points to **Q**



Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)

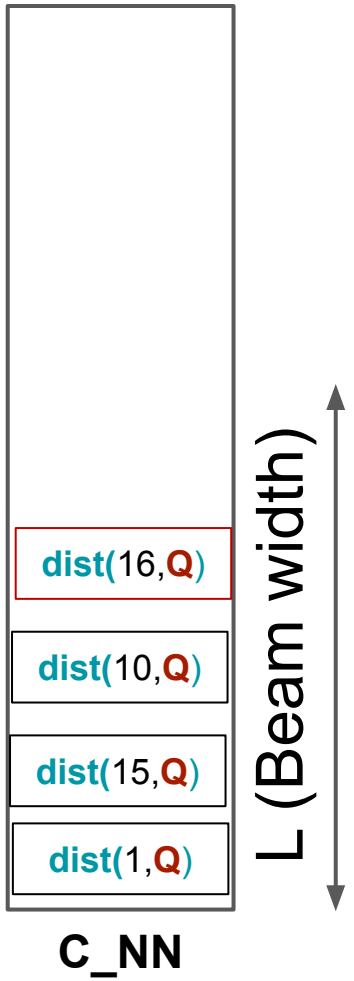


2.1 Select the closest element P from **C_NN**\V

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

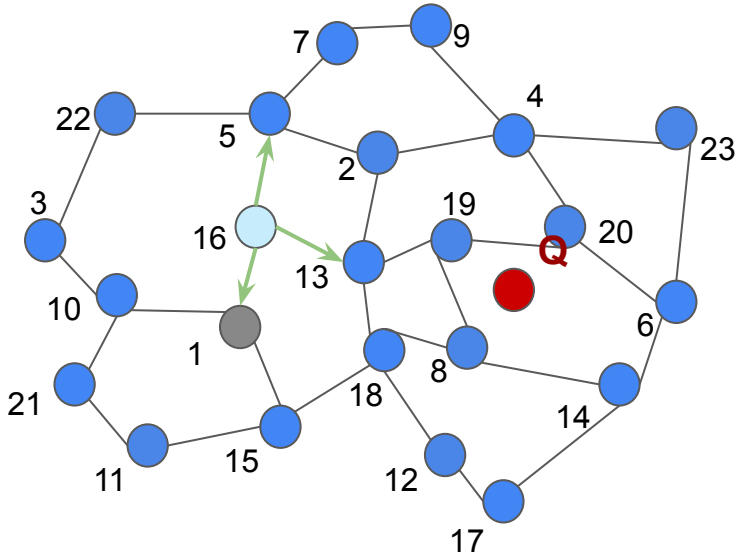
2.4 If |**C_NN**| > **L**: Update **C_NN** to retain closest **L** points to **Q**



Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)

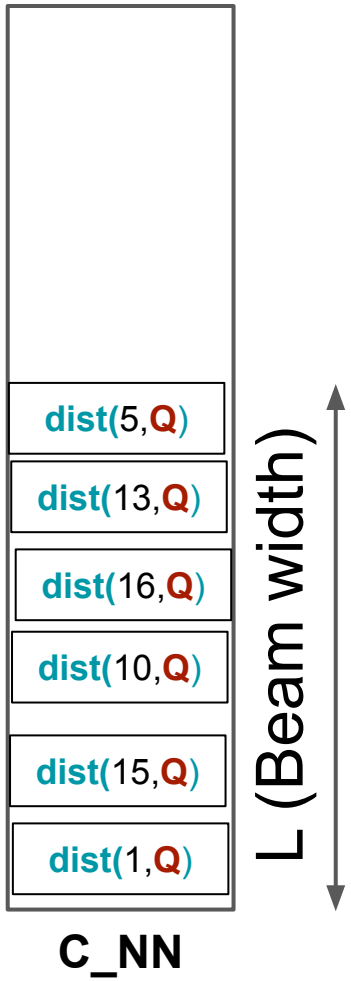


2.1 Select the closest element P from **C_NN**\V

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

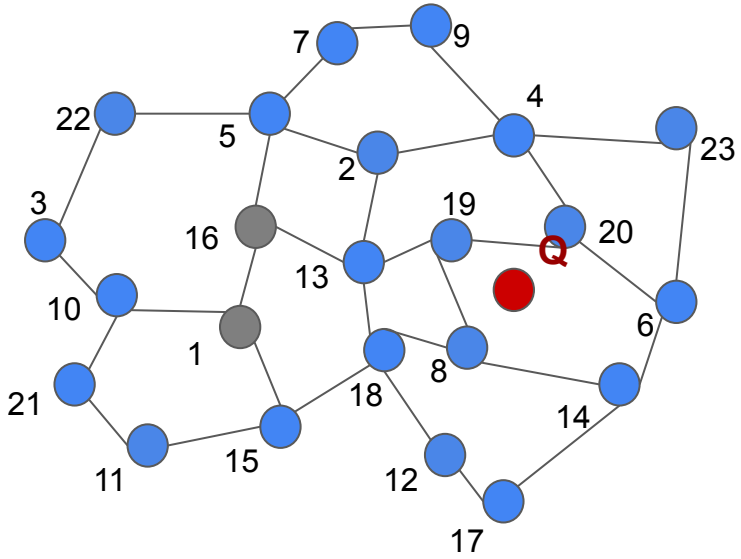
2.4 If |**C_NN**| > **L**: Update **C_NN** to retain closest **L** points to **Q**



Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)

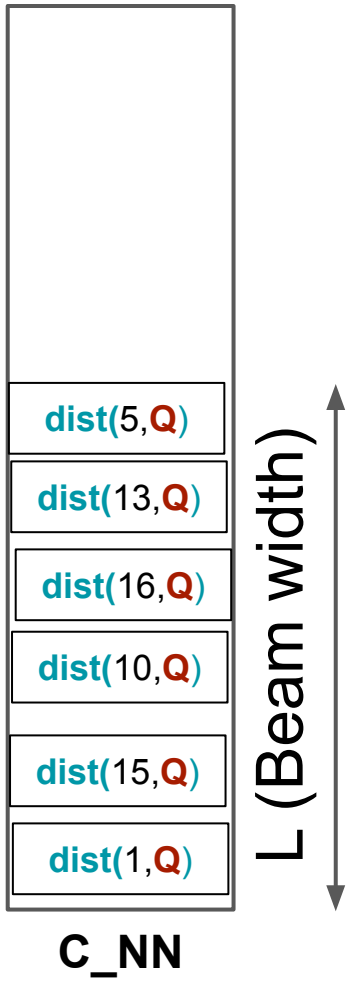


2.1 Select the closest element P from **C_NN**\V

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

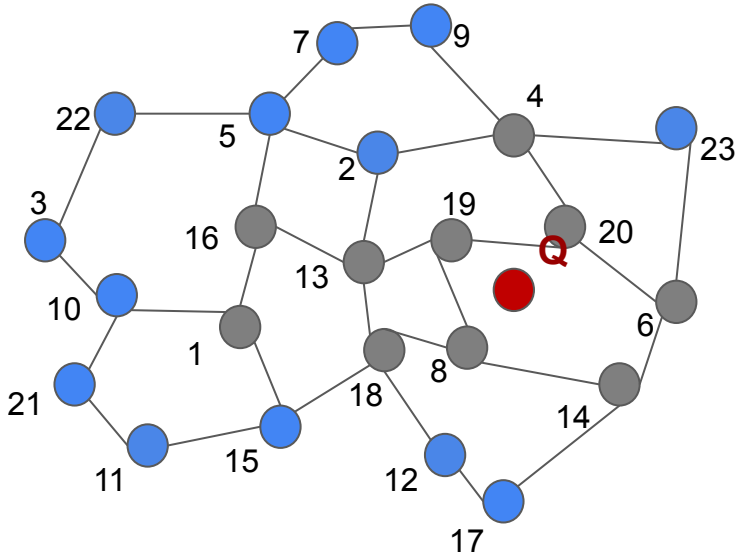
2.4 If |**C_NN**| > **L**: Update **C_NN** to retain closest **L** points to **Q**



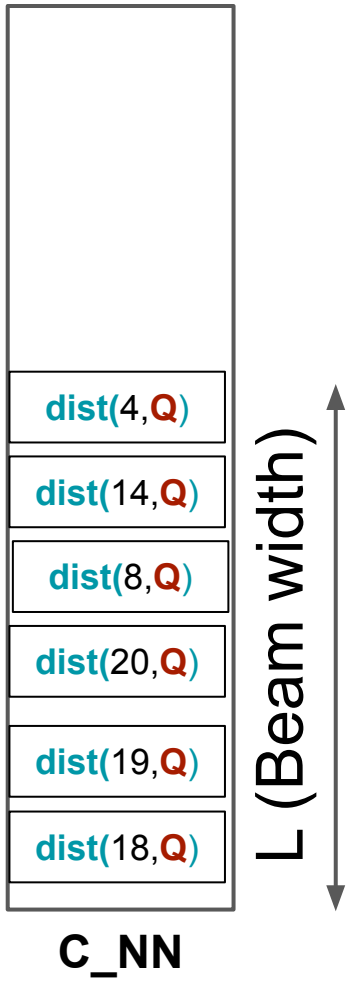
Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

② While(**C_NN**\V≠∅)



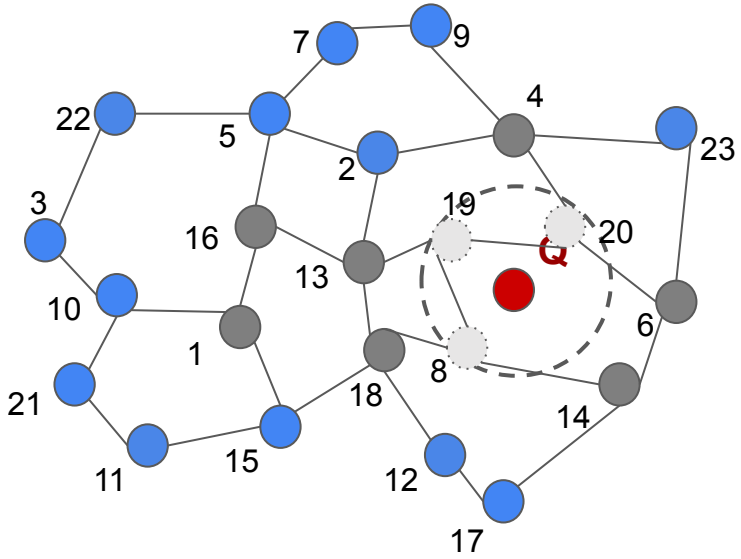
C_NN\V==∅

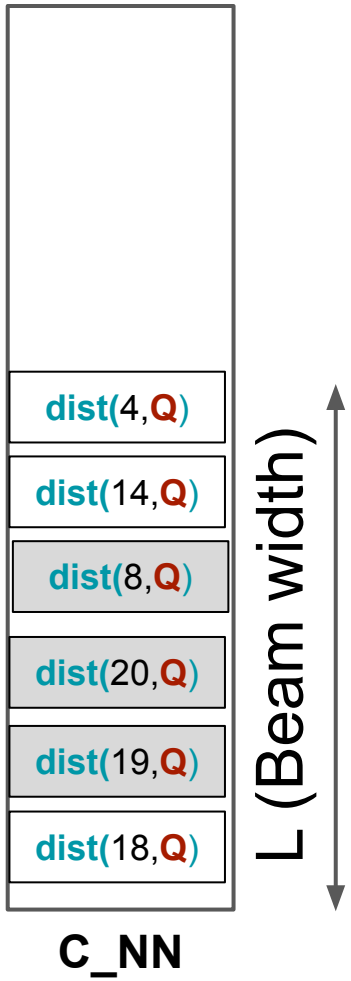


Beam Search

BeamSearch(**G**, **Q**, **entry_node** = 1, K, **L** = 6)

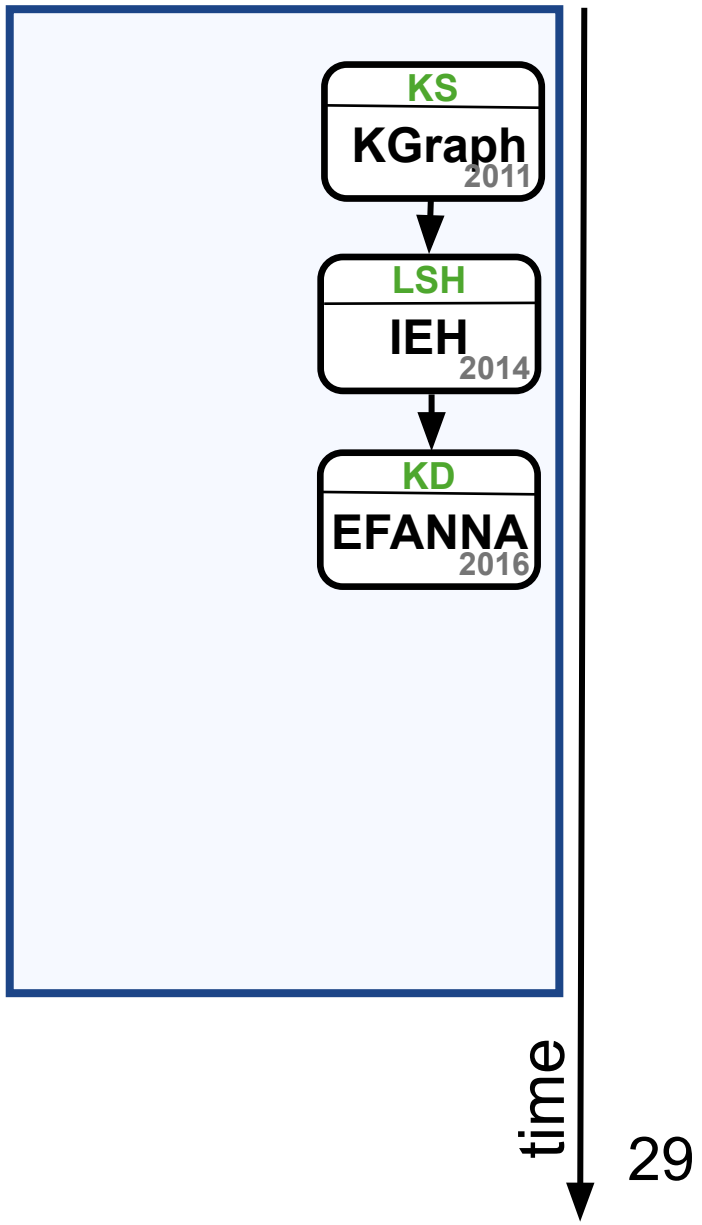
③ Return K closest neighbors from **C_NN**





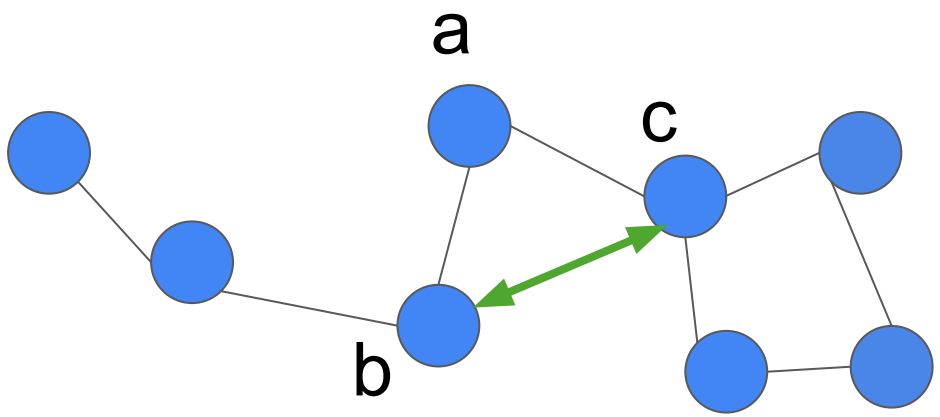
Graph-based Vector Search : New Taxonomy

Neighborhood Propagation based



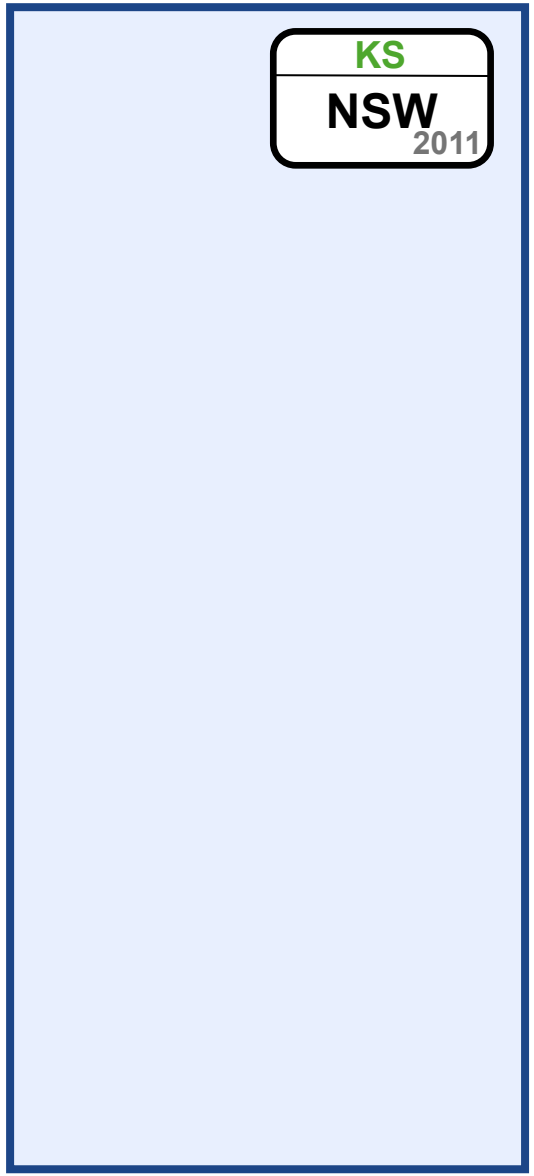
Neighborhood Propagation

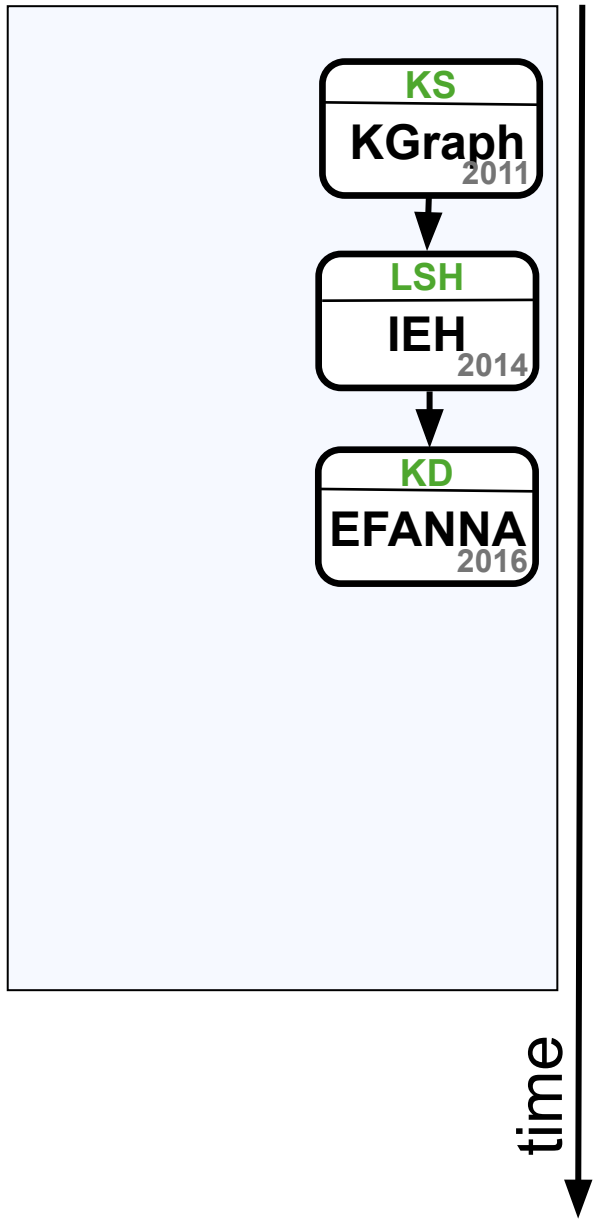
- Neighborhood Propagation through NNDescent



“hi a, your neighbors can become my neighbors too”

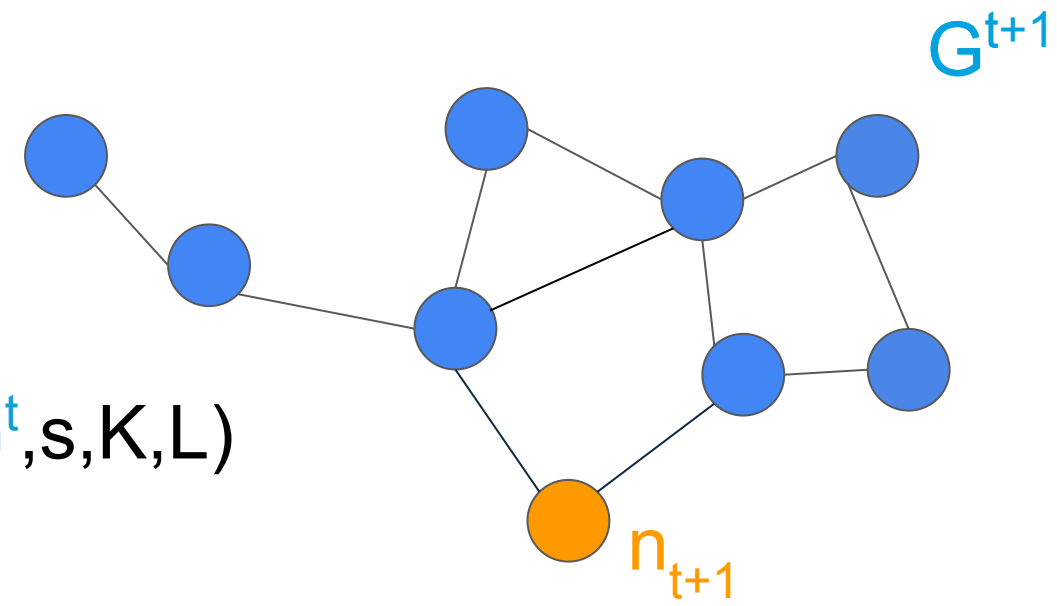
Graph-based Vector Search : New Taxonomy





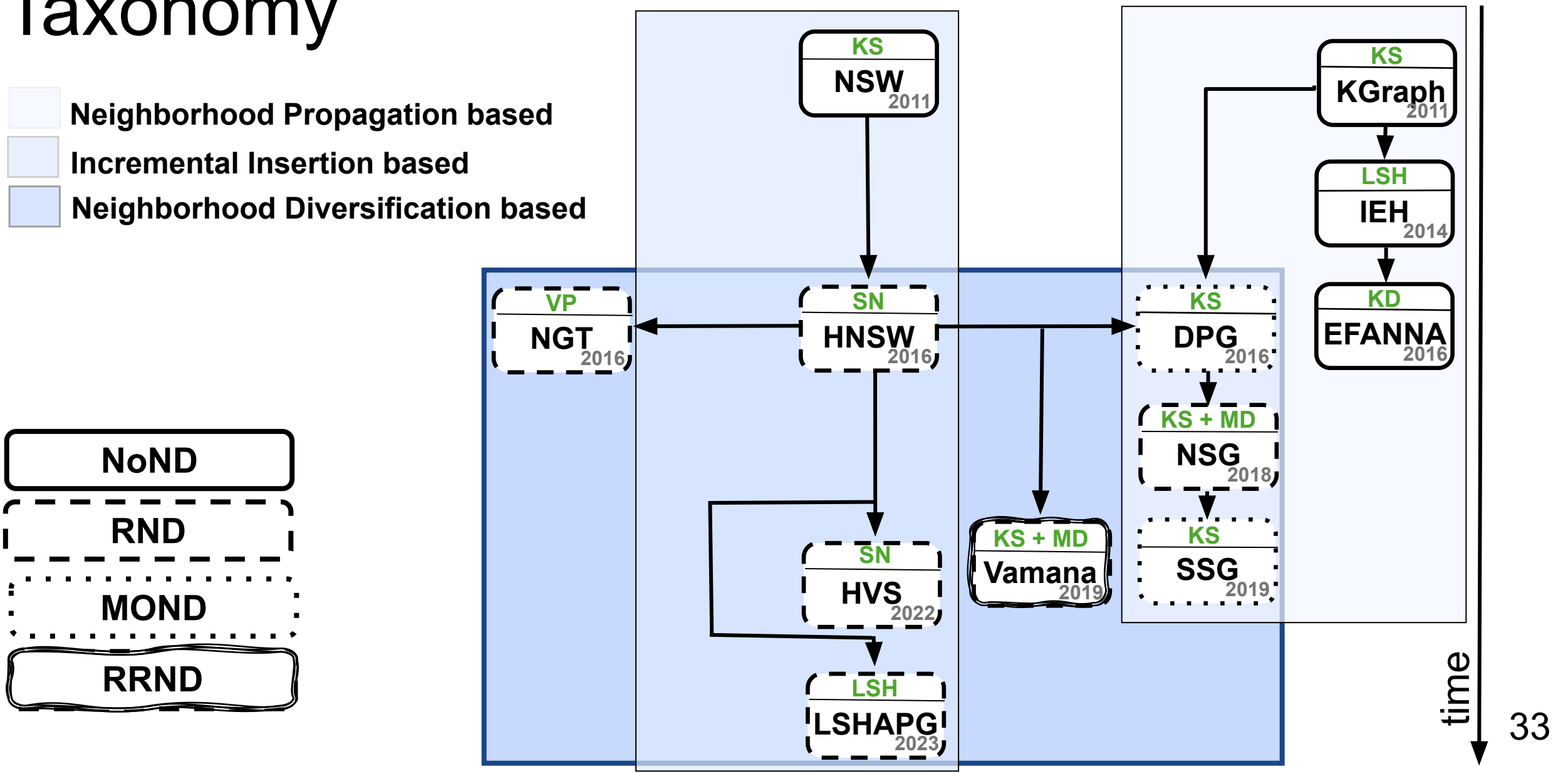
Incremental Insertion

- Insert node incrementally into the graph



BeamSearch(n_{t+1},G^t,s,K,L)

Graph-based Vector Search : New Taxonomy



Relative Neighborhood Diversification

For a node $\mathbf{X_q}$ and a list of candidate neighbors $\mathbf{C_q}$, the node $\mathbf{X_j}$, which belongs to $\mathbf{C_q}$, is selected into the set of $\mathbf{X_q}$'s neighbors $\mathbf{R_q}$ if and only if the following condition holds:

$$\forall \mathbf{X_i} \in \mathbf{R_q},\ \text{dist}(\mathbf{X_q},\mathbf{X_j}) < \text{dist}(\mathbf{X_i},\mathbf{X_j}) \quad \text{(eq1)}$$

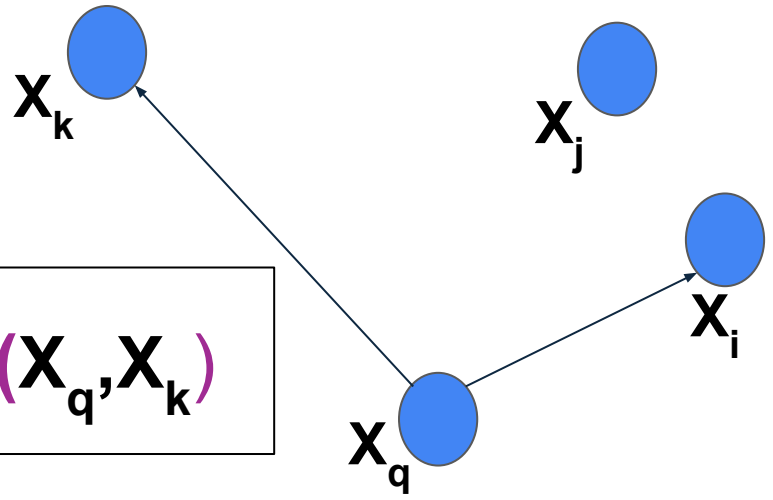
Where:

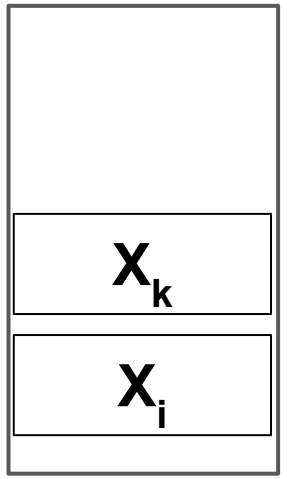
- $\mathbf{X_q}$ is the query node.
- $\mathbf{X_j}$ is a candidate neighbor being considered for inclusion in $\mathbf{R_q}$ and is part of $\mathbf{C_q}$.
- $\mathbf{X_i}$ are nodes already in the set $\mathbf{R_q}$.
- $\text{dist}(\mathbf{X_a},\mathbf{X_b})$ represents the Euclidean distance between nodes $\mathbf{X_a}$ and $\mathbf{X_b}$ in the d-dimensional space.

Relative Neighborhood Diversification

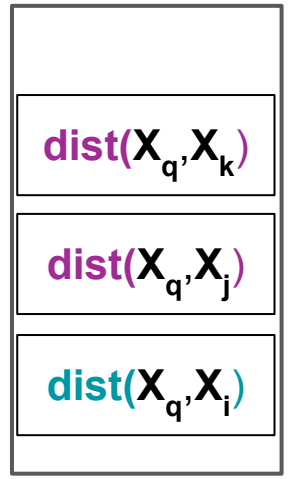
$dist(X_i,X_j) < dist(X_q,X_j)$

$dist(X_i,X_k) > dist(X_q,X_k)$





R_q

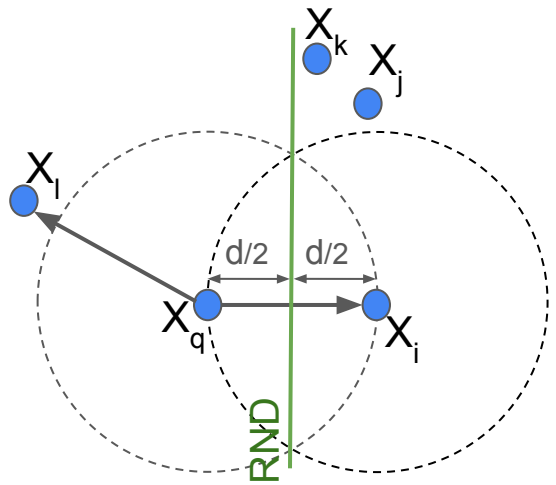


C_q

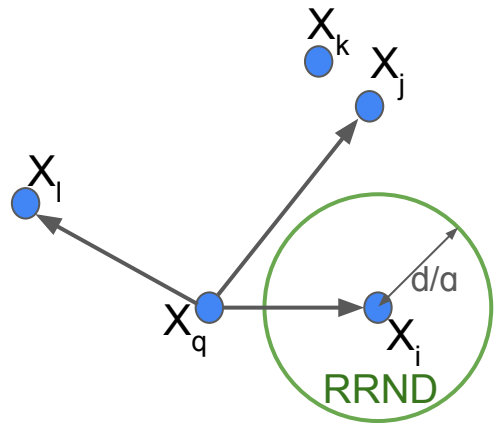
Neighborhood Diversification in the SOTA GANNS methods

Neighborhood Diversification (ND) approaches in literature:

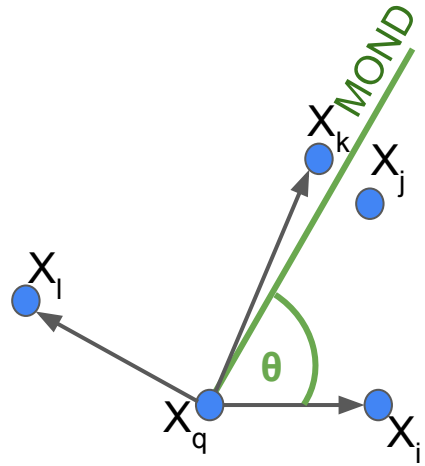
1. Relative ND (RND)
2. Relaxed RND (RRND): $(eq_1) \Rightarrow \forall \mathbf{X_j} \in \mathbf{R_q}$, $dist(\mathbf{X_q},\mathbf{X_i}) < \alpha.dist(\mathbf{X_i},\mathbf{X_j})$ for $\alpha > 1$
3. Maximum-Oriented ND (MOND): $(eq_1) \Rightarrow \forall \mathbf{X_j} \in \mathbf{R_q}$, $\cos(\angle \mathbf{X_i X_q X_j}) < \cos(\theta)$



RND

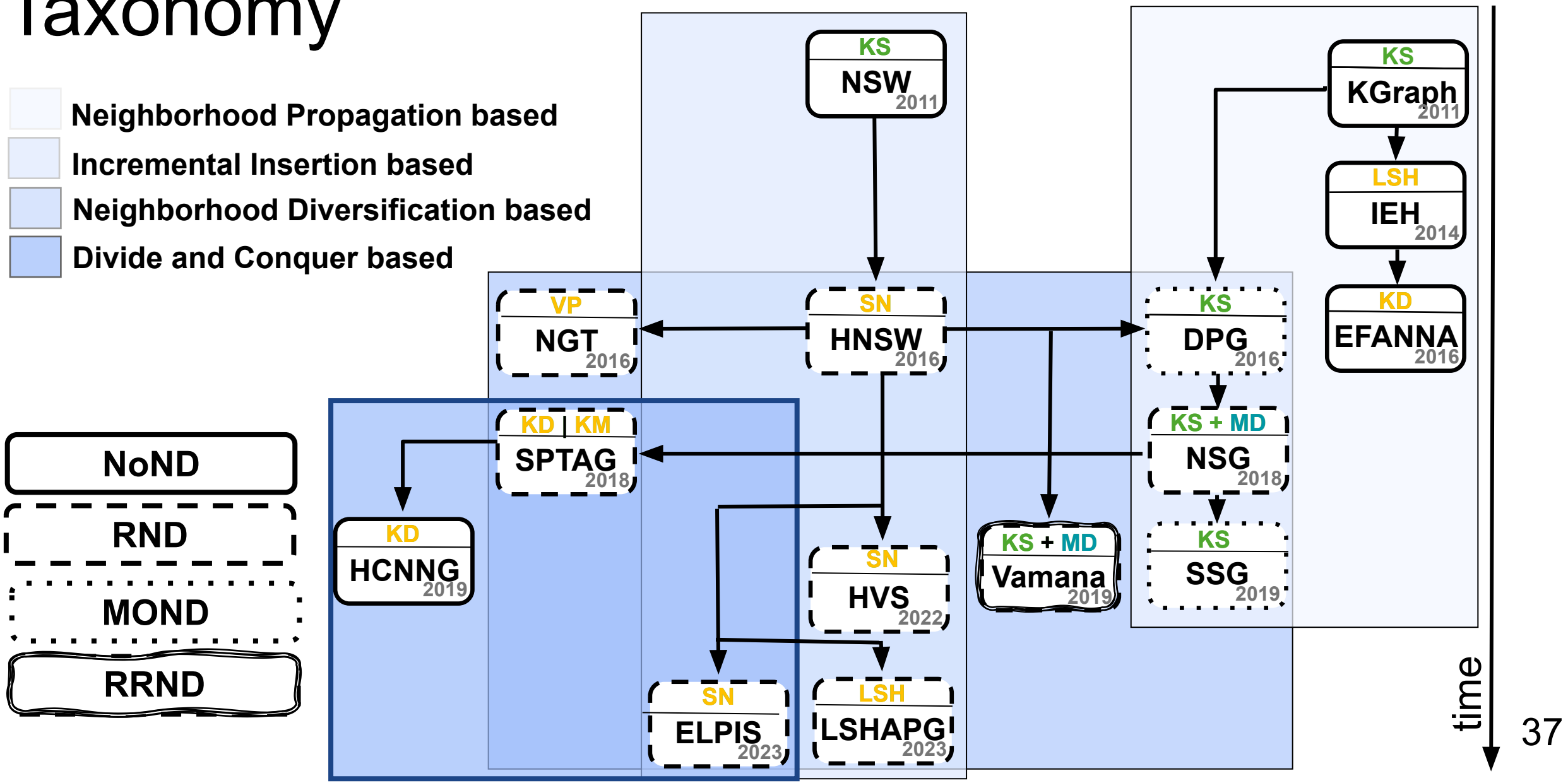


RRND



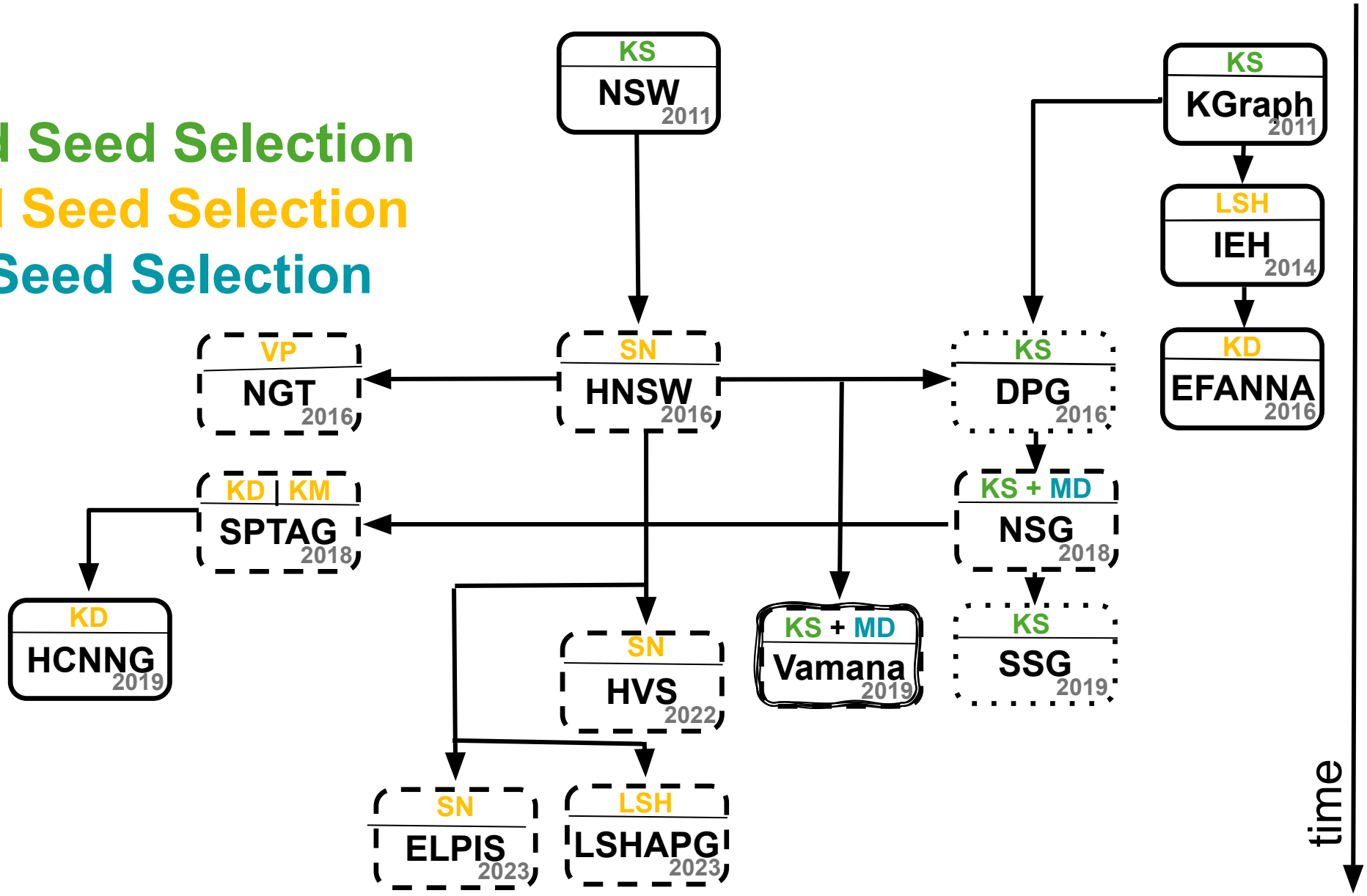
MOND

Graph-based Vector Search : New Taxonomy



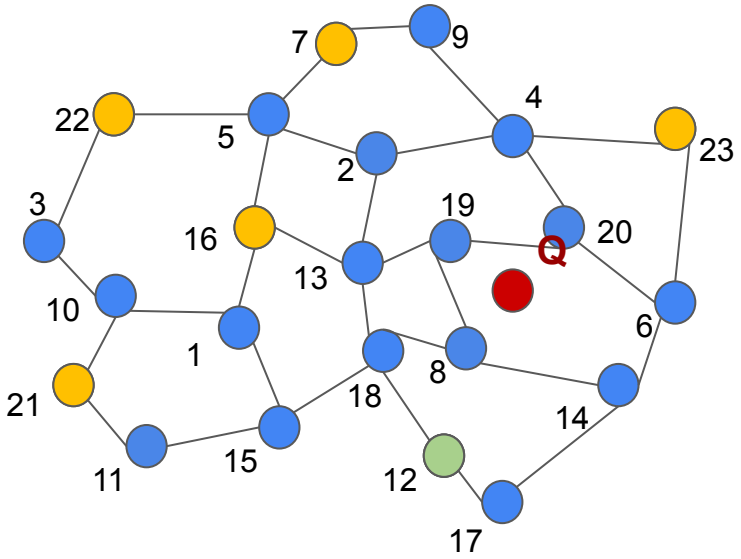
Graph-based Vector Search : Seed Selection

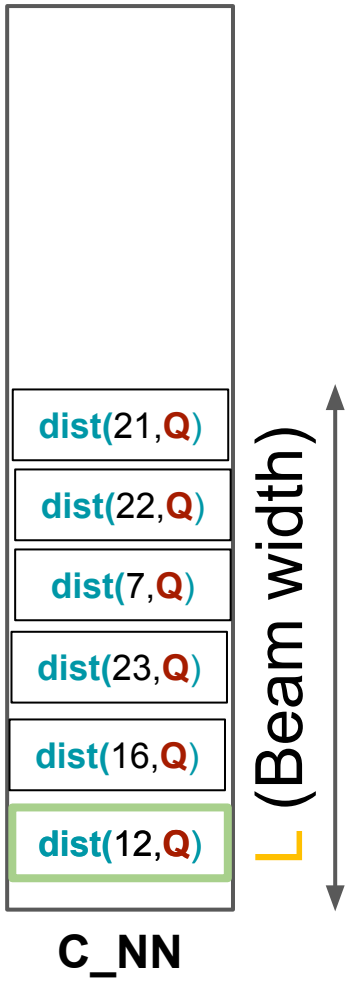
- **Randomized Seed Selection**
- **Index-based Seed Selection**
- **Predefined Seed Selection**



Randomized Seed Selection

1. Select up to L random seed points
2. Warm the priority queue with seed points
3. Select the closest point to the query as entry node



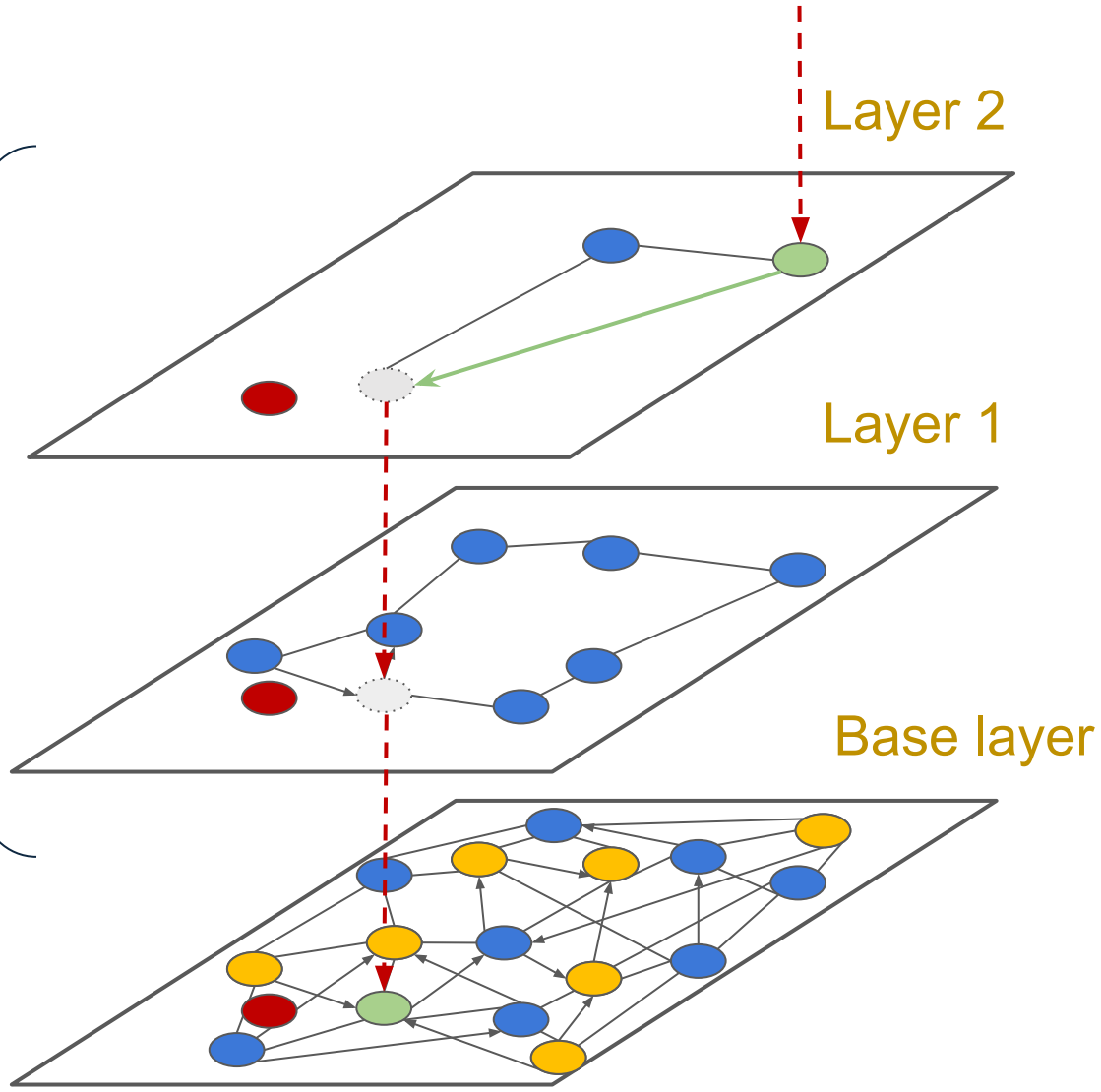


Index-based Seed Selection

1. Construct one or more lightweight indices over a representative sample of the dataset.
   a. Stacked NSW (SN)
   b. KD-Trees (KD)
   c. K-mean balanced Trees (KM)
   d. VP Trees (VP)
   e. LSH

Stacked NSW (SN)

Hierarchical Layers
(Stacked NSW)



Index-based Seed Selection

1. Construct one or more lightweight indices over a representative sample of the dataset.
   a. Stacked NSW (SN)
   b. KD-Trees (KD)
   c. K-mean balanced Trees (KM)
   d. VP Trees (VP)
   e. LSH
2. Retrieve one or multiple seed nodes from the index to initialize the Beam Search

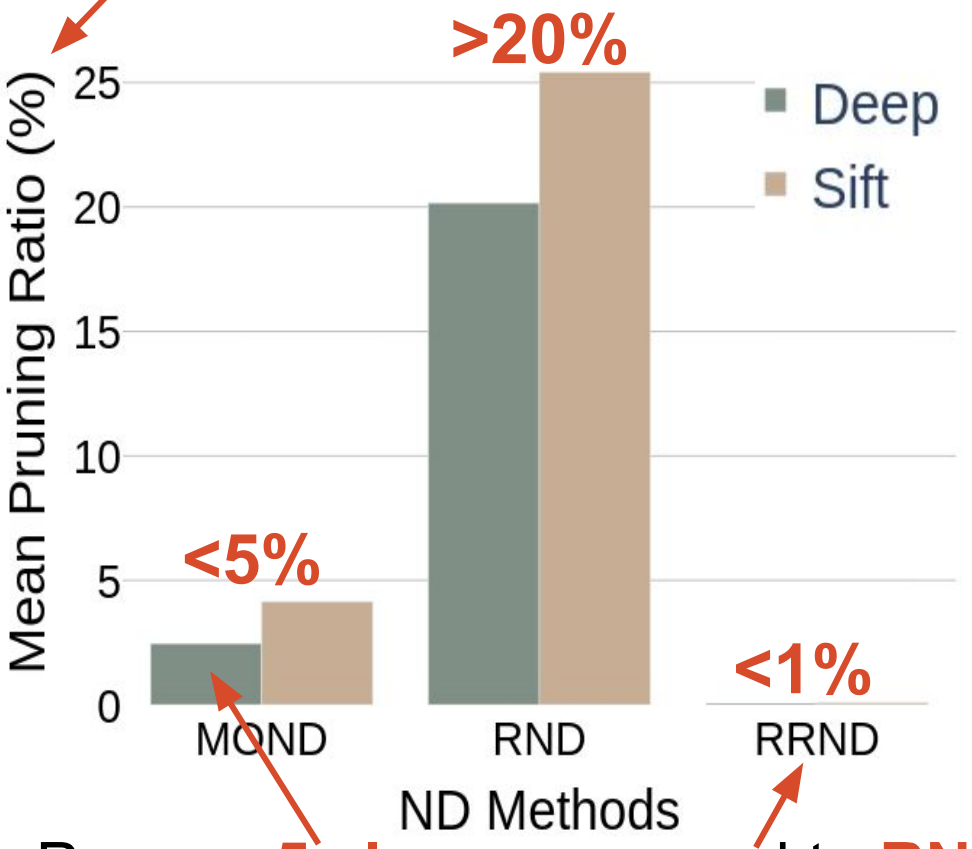
Comprehensive Experimental Evaluation

Datasets

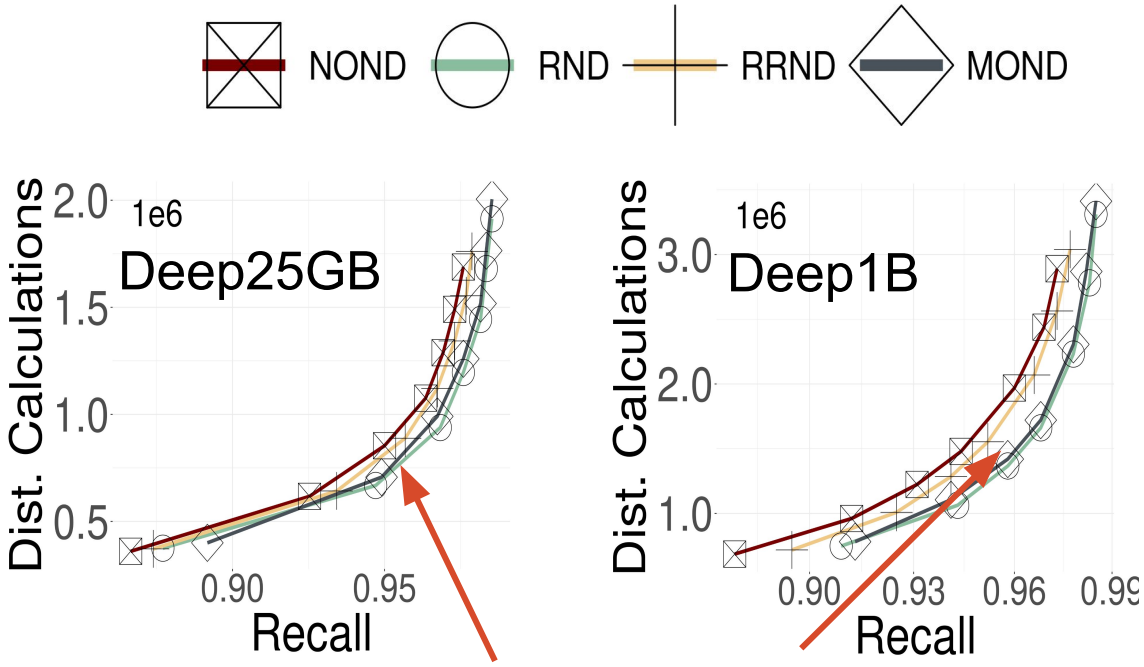
- **Sift1B**: 1 billion vectors of 128 dimensions representing the Sift image feature descriptors.
- **Deep1B**: 1 billion vectors of 96 dimensions extracted from the last layers of a convolutional neural network.
- **Sald**: 200 million neuroscience MRI data series of size 128 .
- **Seismic**: 100 million data series of size 256 representing earthquake recordings at seismic stations worldwide.
- **Gist**: 1 million vectors of 960 dimensions representing image descriptors that capture spatial structure and color layout.
- **ImageNet**: 1 million vectors of 256 dimensions generated from ImageNet using a ResNet50 model, followed by PCA for dimensionality reduction.
- **Text-to-Image**: 1 billion 200-dimensional image embeddings (from Se-ResNext-101) paired with 50 million text queries (from DSSM) for cross-modal retrieval tasks.
- **RandPow*i***: contains vectors of 256 dimensions generated randomly following power law distribution using power law exponent *i*.

Neighborhood Diversification

Smaller Index size and **Search Memory Footprint**

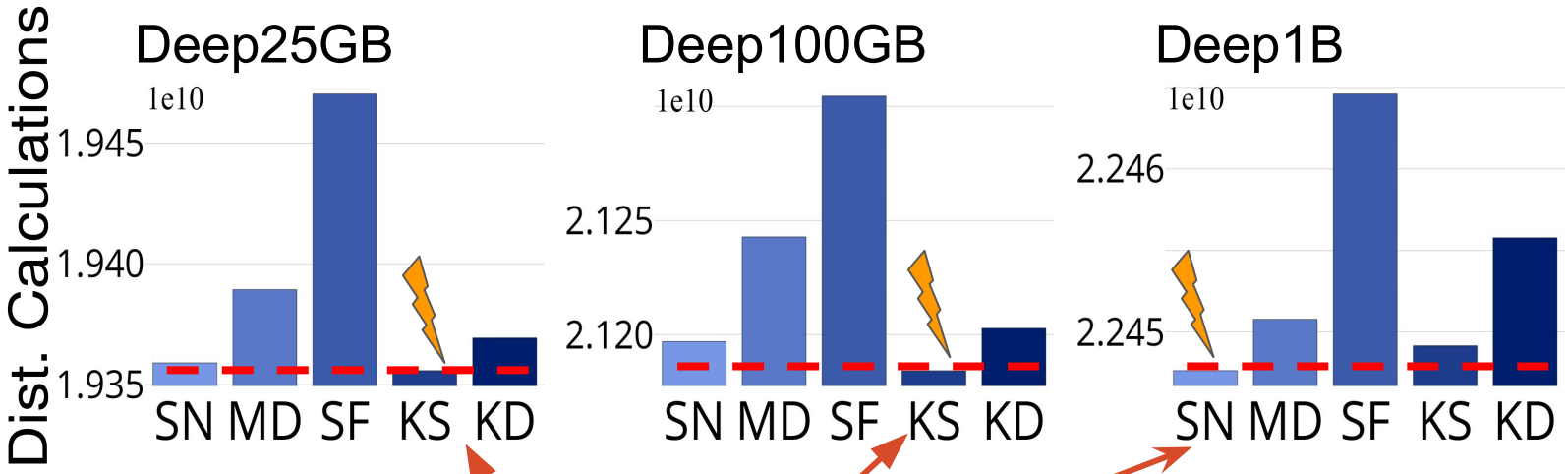


Prunes **5x less** compared to **RND**



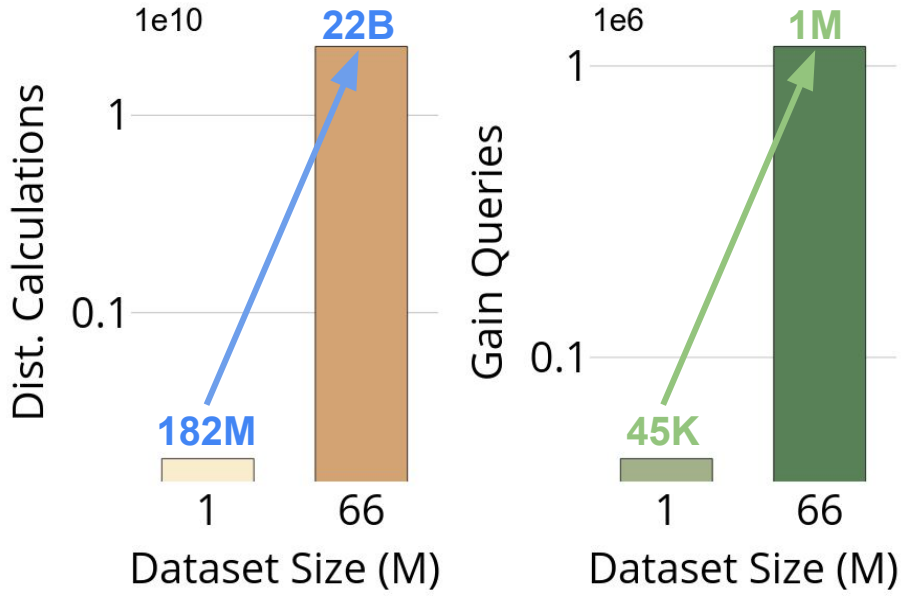
RND gives the best **Search Performance**

Seed selection



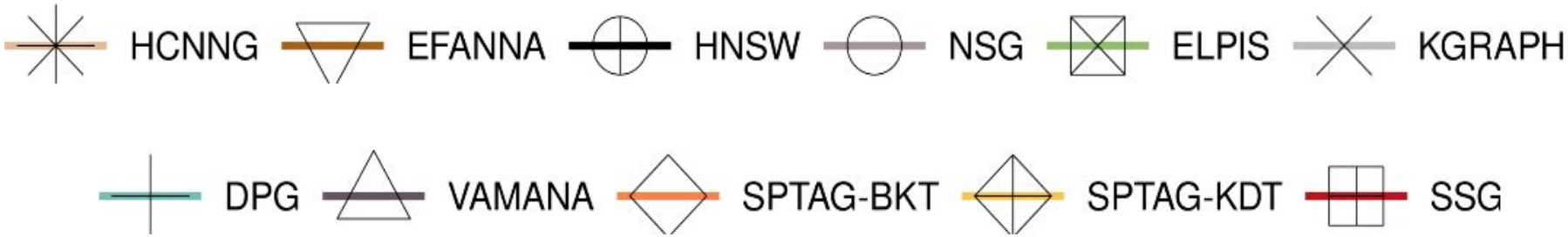
Optimizing **data structures** for **seed selection** enhances **Search Efficiency** on **extensive datasets**

The choice of seed selection impacts **Index Efficiency** as well



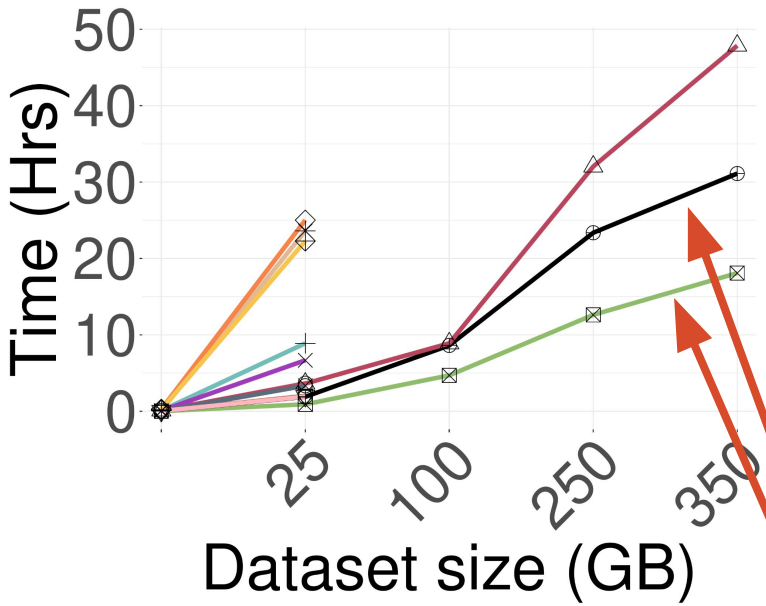
Sota Graph-based Vector Search Methods

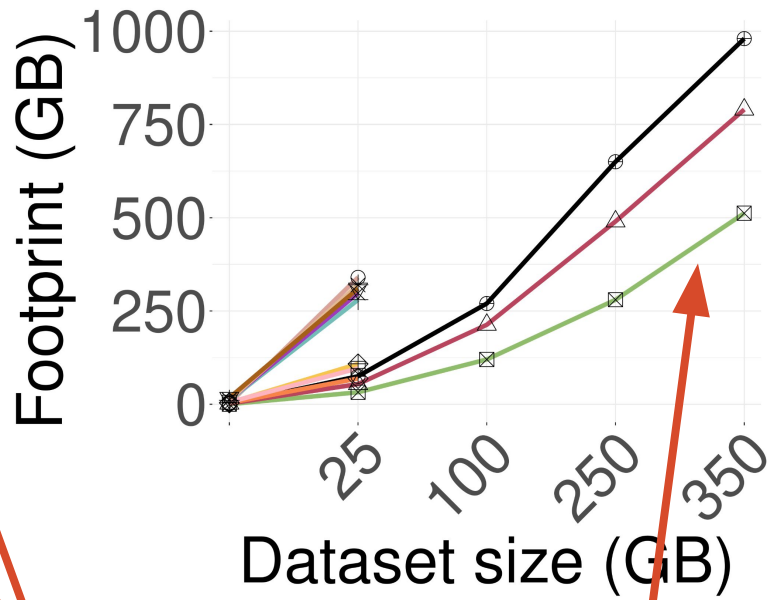
- HNSW [12]
- NSG [10]
- VAMANA [13]
- SPTAG [14]
- NGT [15]
- SSG [9]
- LSH-APG [17]
- HCNNG [18]
- DPG [11]
- EFANNA [8]
- KGRAPH [7]

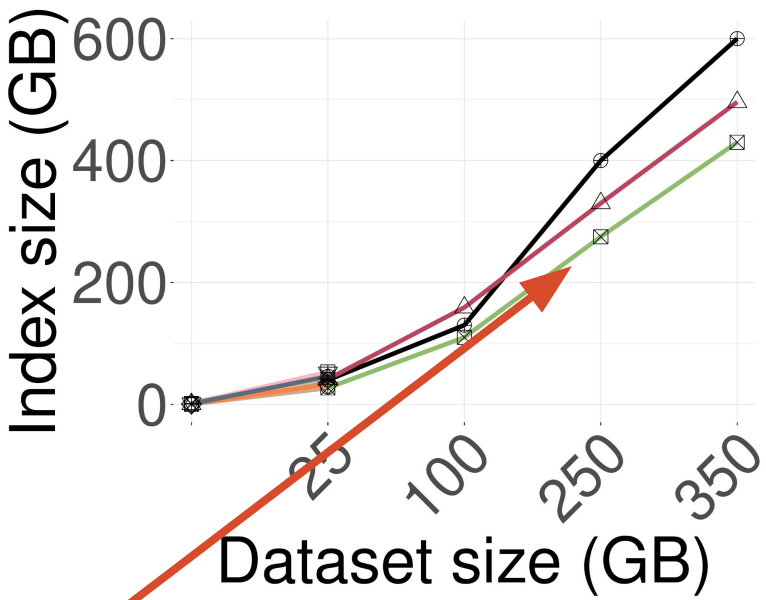


Indexing Performance

Deep1B Dataset



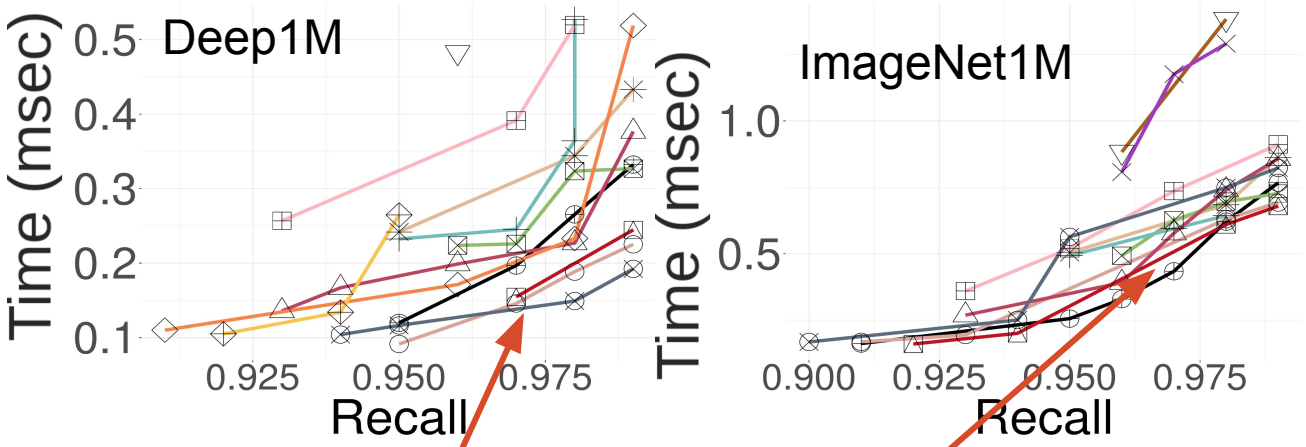




Incremental Insertion based Graph Methods

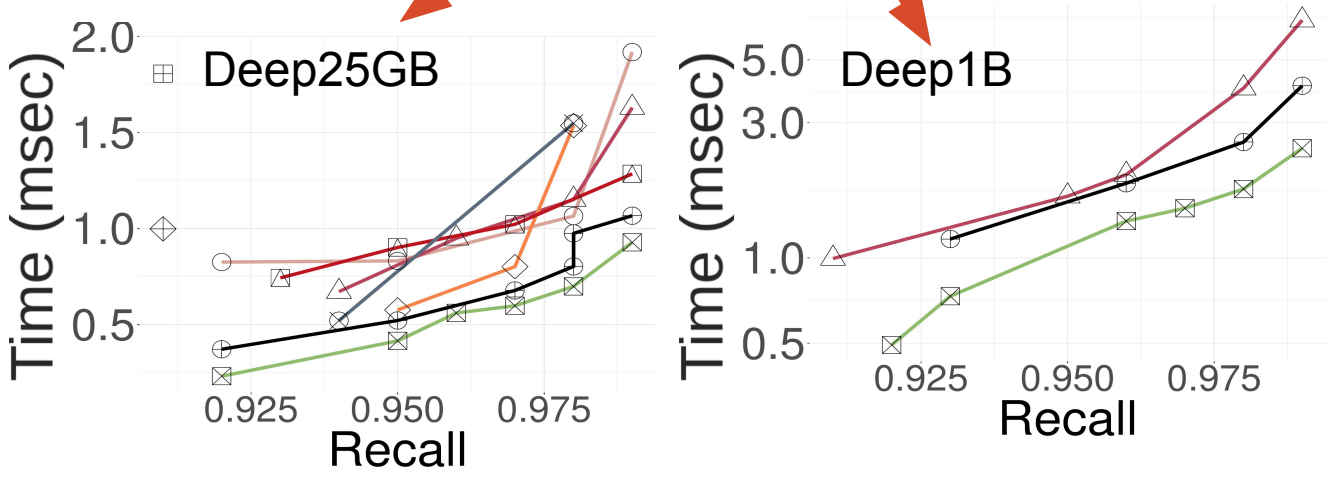
are the most **Scalable**

Search Performance

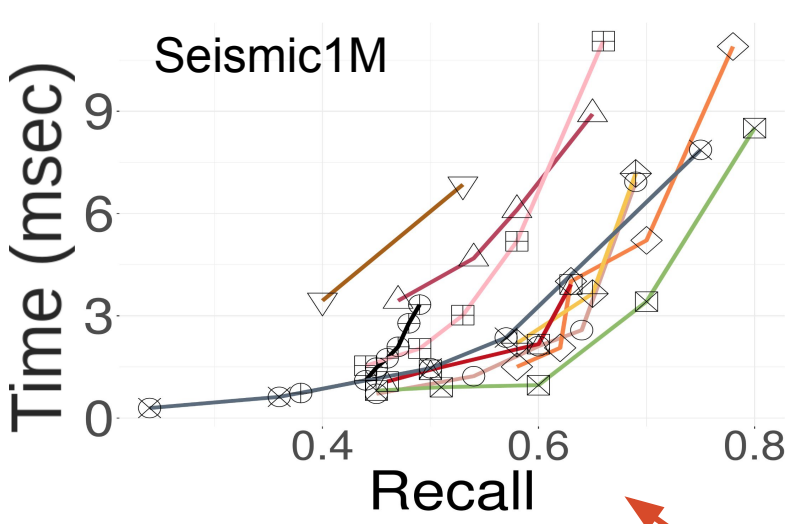


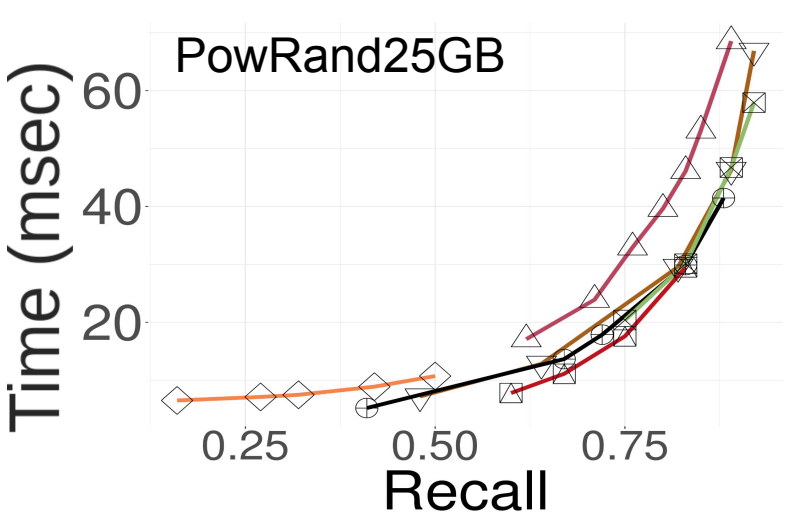
Most GANNS approaches Fails to Scale to **Large Dataset Efficiently**

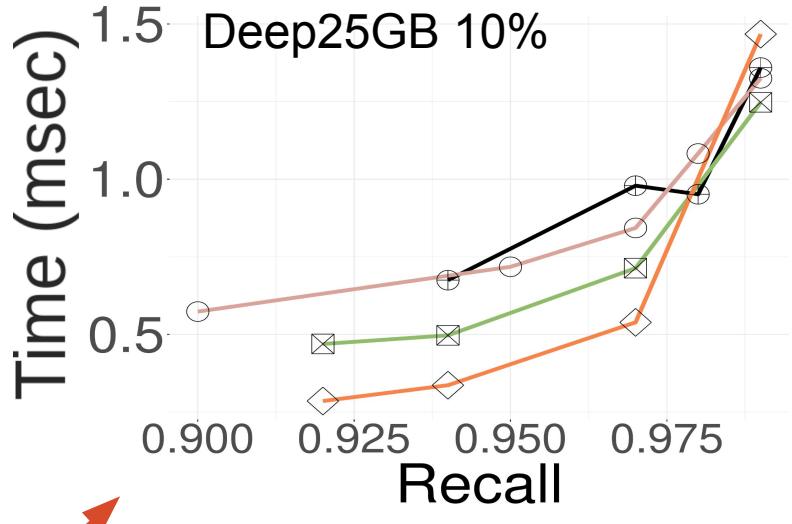
Neighborhood Diversification graph-based methods gives the best **Search Performance**



Search Performance







Divide and Conquer graph-based methods gives the best

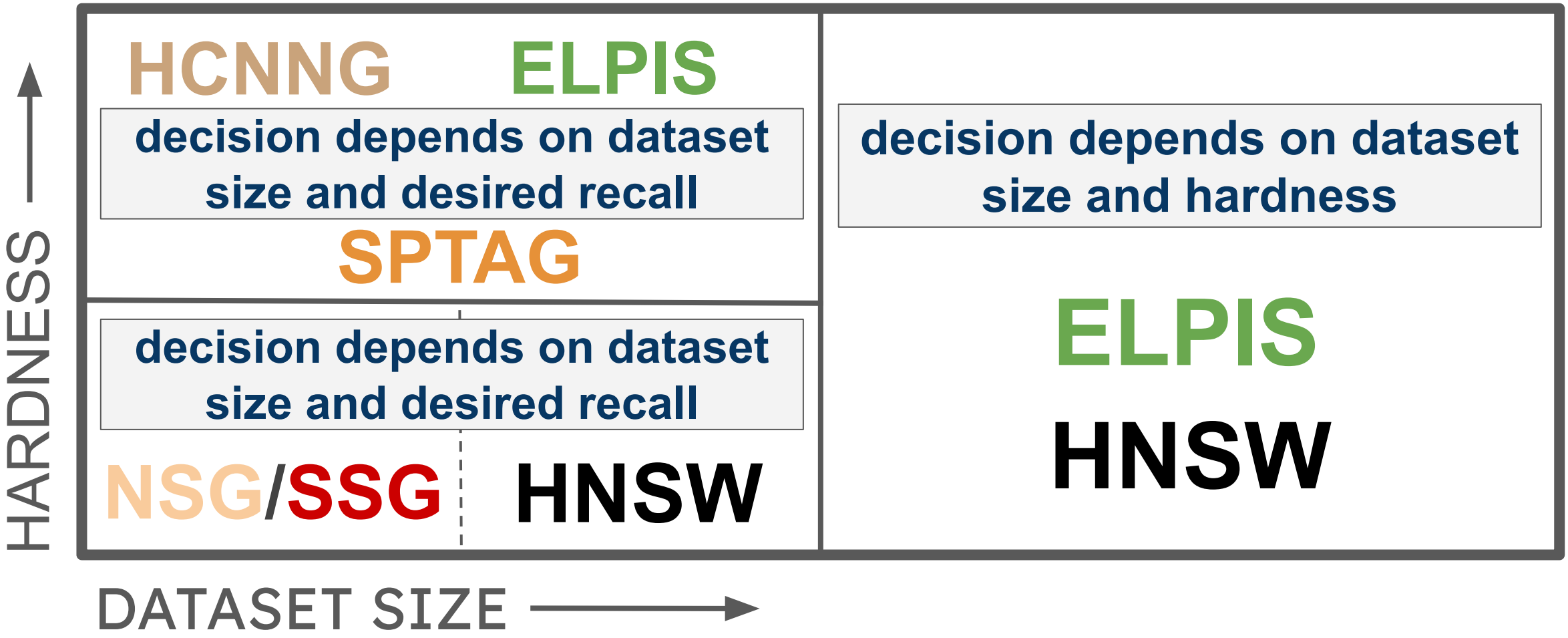
Search Performance on Hard Datasets and workload

Recommendations

✓ Good ~ Medium × Bad

Method	Query Answering			Index Building		
	Efficiency	Accuracy	Tuning	Efficiency	Footprint	Tuning
HNSW	✓	✓	✓	✓	✓	✓
ELPIS	✓	✓	~	✓	✓	~
VAMANA	✓	✓	✓	✓	✓	~
NSG	✓	✓	✓	~	~	~
SSG	✓	✓	✓	~	~	~
EFANNA	×	~	×	×	×	×
KGRAPH	×	×	×	×	×	×
DPG	×	~	~	~	~	~
SPTAG	~	✓	×	×	✓	×
HCNNG	✓	✓	✓	✓	✓	~
LSHAPG	×	~	×	~	✓	✓
NGT	~	~	×	×	✓	×
SPTAG	~	~	~	×	✓	×

Recommendations



Thank you

check our other SIGMOD'25 papers on high-d vector similarity search:

- **LeaFi:** Data Series Indexes on Steroids with Learned Filters
  - pruning based on machine learning leads to 32x faster query answering
- **RWalks:** Random Walks as Attribute Diffusers for Filtered Vector Search
  - general graph-based filtered vector search for fast filtered (2x faster) and unfiltered vector search (13x faster)
- **Subspace Collision:** An Efficient and Accurate Framework for High-dimensional Approximate Nearest Neighbor Search
  - subspace indexing leads to 100x faster query answering with the same quality guarantees

Code Available @ github.com/iliasazizi/GVS

Contact: azizii.ilias@gmail.com

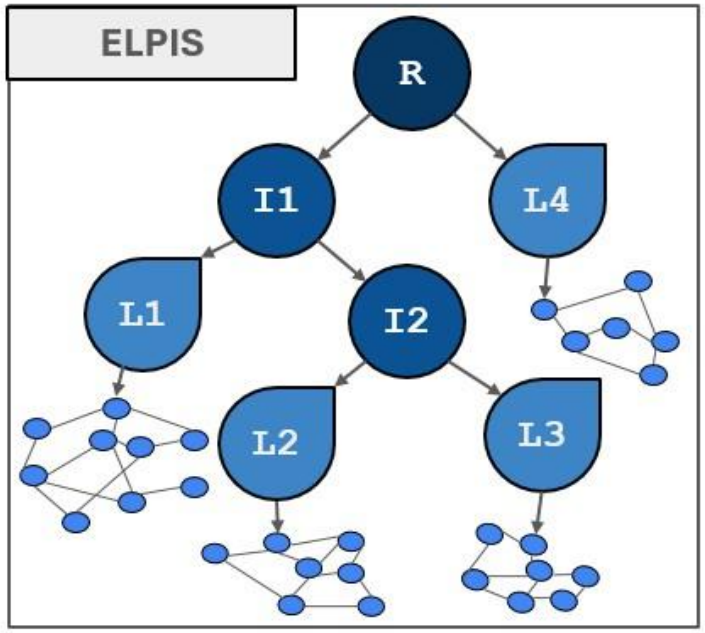
References

1. Ilias Azizi, Karima Echihabi, Themis Palpanas. Elpis: Graph-Based Similarity Search for Scalable Data Science. PVLDB 2023.
2. Ilias Azizi, Vector Search on Billion-Scale Data Collections. In VLDB PhD Workshop, 2024.
3. Ilias Azizi, Karima Echihabi, Themis Palpanas. Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. SIGMOD 2025.
4. Wang, Y., Wang, P., Pei, J., Wang, W., & Huang, S. A data-adaptive and dynamic segmentation index for whole matching on time series. PVLDB 2013.
5. Echihabi, K., Fatourou, P., Zoumpatianos, K., Palpanas, T., & Benbrahim, H. Hercules Against Data Series Similarity Search. PVLDB 2022
6. Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems, 45:61–68, 2014.
7. W. Dong. Kgraph, an open source library for k-nn graph construction and nearest neighbor search. www.kgraph.org, 2022.
8. C. Fu and D. Cai. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. arXiv preprint arXiv:1609.07228, 2016.
9. C. Fu, C. Wang, and D. Cai. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.
10. C. Fu, C. Xiang, C. Wang, and D. Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. Proc. VLDB Endow., 12(5):461–474, 2019.
11. W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. 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.
12. Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell., 42(4):824–836, 2020.
13. S. J. Subramanya, R. Kadekodi, R. Krishaswamy, and H. V. Simhadri. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 13766–13776, 2019.
14. Q. Chen, H. Wang, M. Li, G. Ren, S. Li, J. Zhu, J. Li, C. Liu, L. Zhang, and J. Wang. SPTAG: A library for fast approximate nearest neighbor search, 2018.
15. Yahoo Japan Corporation. Ngt: Neighborhood graph and tree for high-dimensional data. https://github.com/yahoojapan/NGT, 2022. Accessed: 2024-10-20
16. J. V. Munoz, M. A. Gonçalves, Z. Dias, and R. d. S. Torres. Hierarchical clustering-based graphs for large scale approximate nearest neighbor search. Pattern Recognition, 96:106970, 2019.
17. X. Zhao, Y. Tian, K. Huang, B. Zheng, and X. Zhou. Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. Proceedings of the VLDB Endowment, 16(8):1979–1991, 2023.
18. K. Lu, M. Kudo, C. Xiao, and Y. Ishikawa. Hvs: hierarchical graph structure based on voronoi diagrams for solving approximate nearest neighbor search. Proceedings of the VLDB Endowment, 15(2):246–258, 2021.

ELPIS: Latency-Optimized Graph-Based Similarity Search

ELPIS

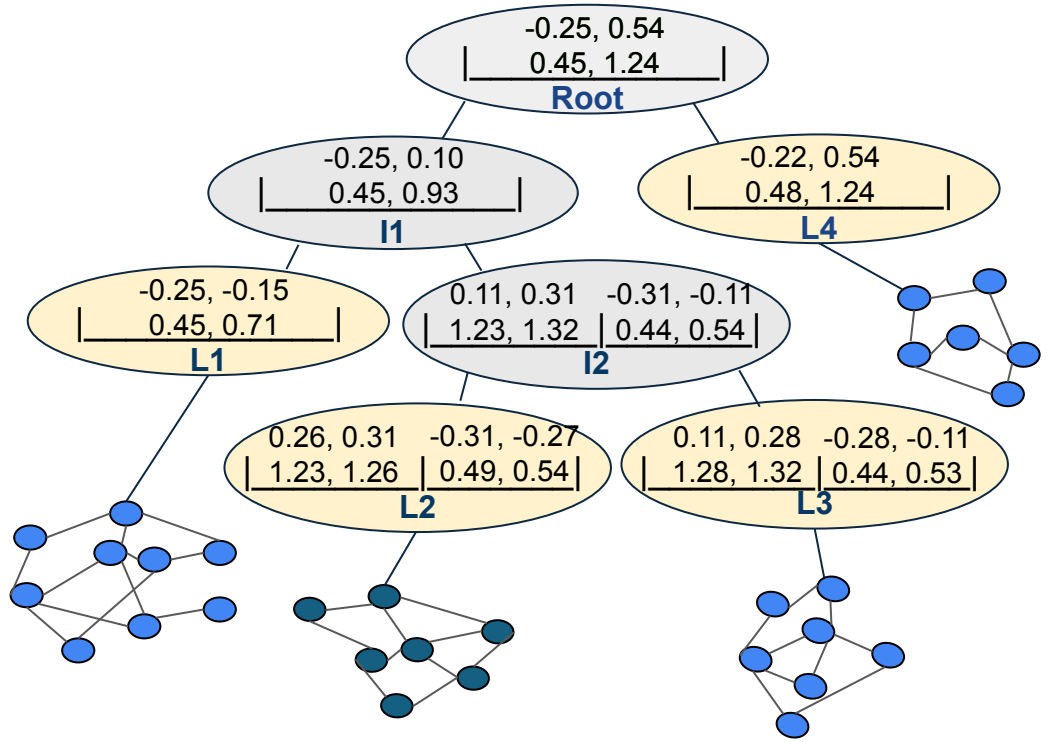
- Tree + Graph based approach for In-memory ng-approximate similarity search
  - Indexing Scalability
  - Search latency efficiency
  - Search accuracy on hard datasets



ANN Family	Advantage	Disadvantage
Tree-Based (data series)	**Low index construction time** **Can support different flavors of search**	**Low search performance on hard query workloads**
Hash-Based	Support theoretical guarantees on query efficiency and accuracy	High index construction time High footprint Low search performance (efficiency, accuracy)
Graph-Based	**Excellent empirical query efficiency**	**High Index construction time** **High footprint** **Low search accuracy on hard dataset**

ELPIS

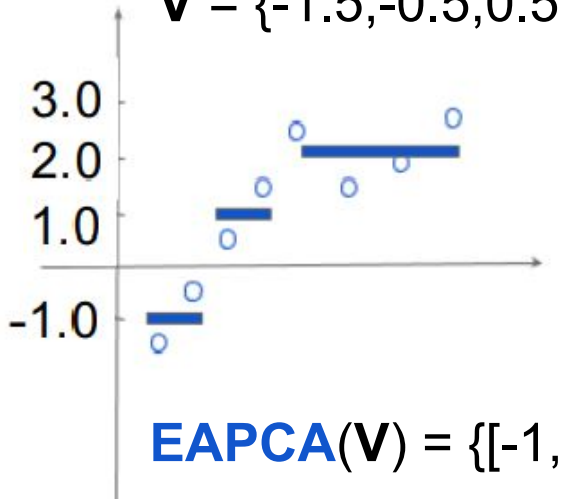
- **ELPIS** combines tree and graph structures for efficient in-memory ng-approximate vector similarity search.



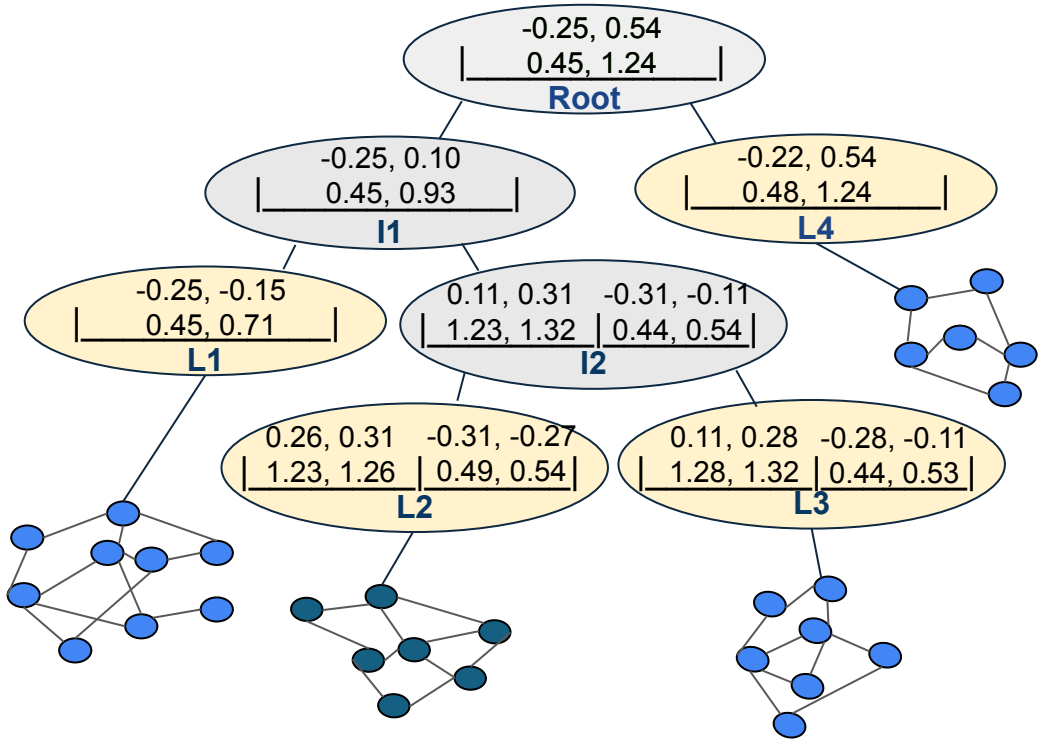
ELPIS: Indexing

- **EAPCA** data series summarization

V = {-1.5,-0.5,0.5,1.5,2.5,1.5,2,2.6}

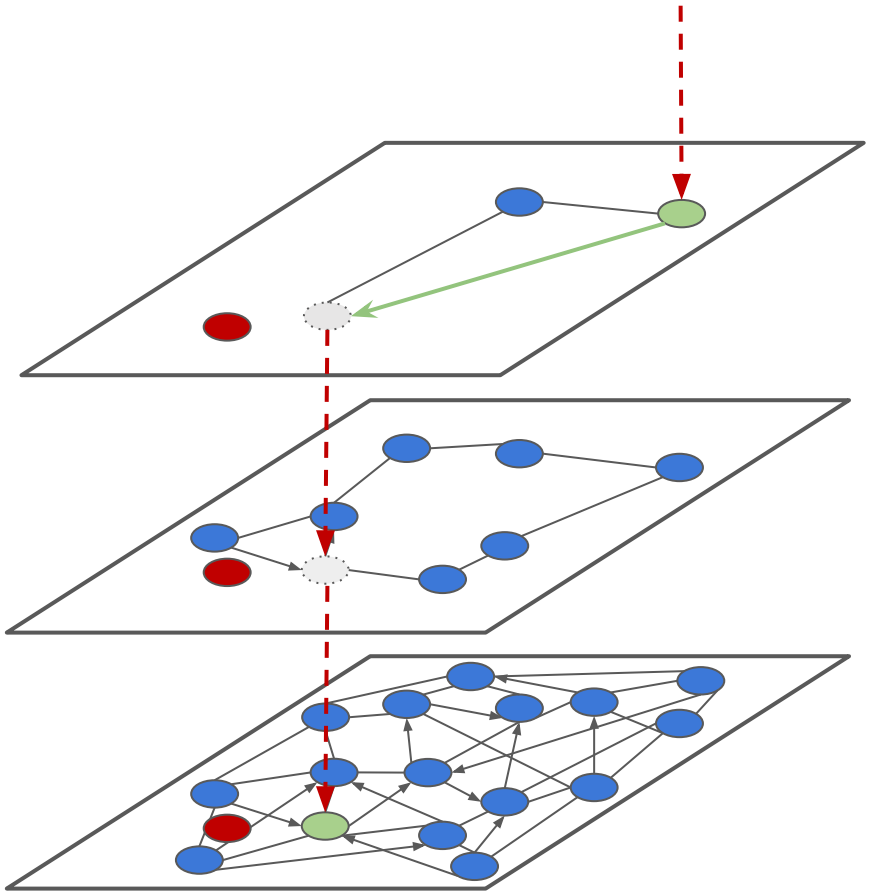


EAPCA(**V**) = {[-1,0.5],[1,0.5],[2.15,0.44]}



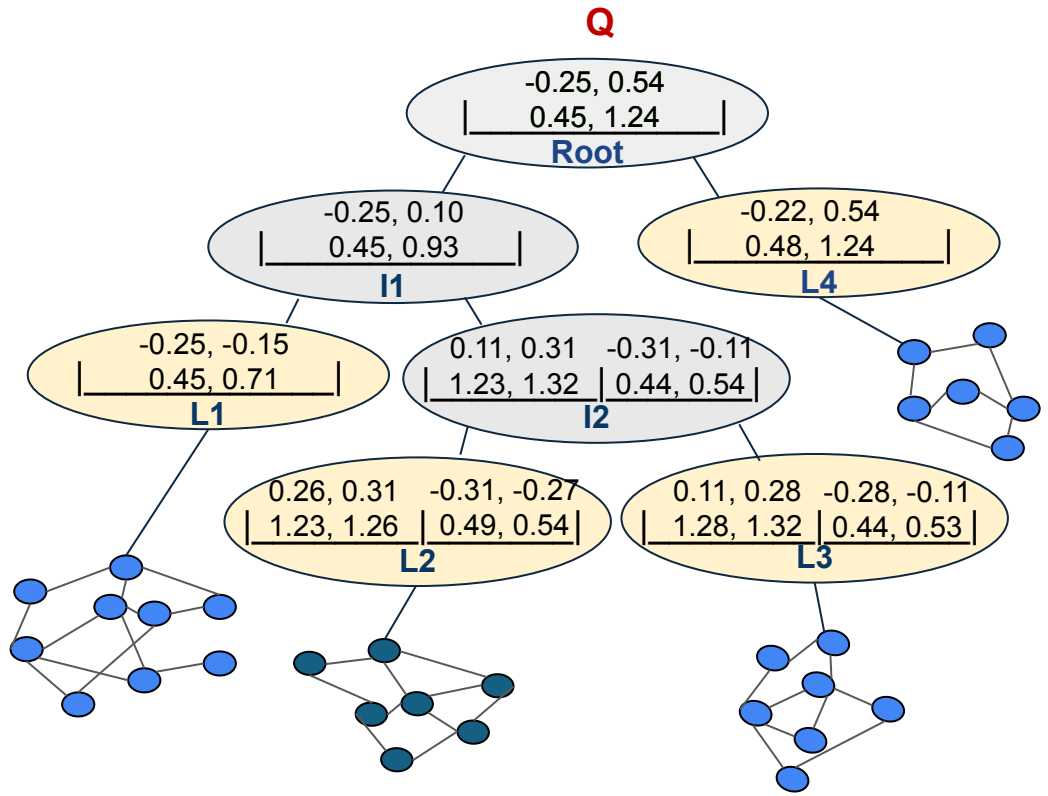
ELPIS: Leaf Indexing

- HNSW stack multiple NSW into a mutli-resolution layers of approximate NSW graphs to solve connectivity problem
- HNSW's hierarchical layer edges act as long-range connections, aiding distance calculation pruning in searches



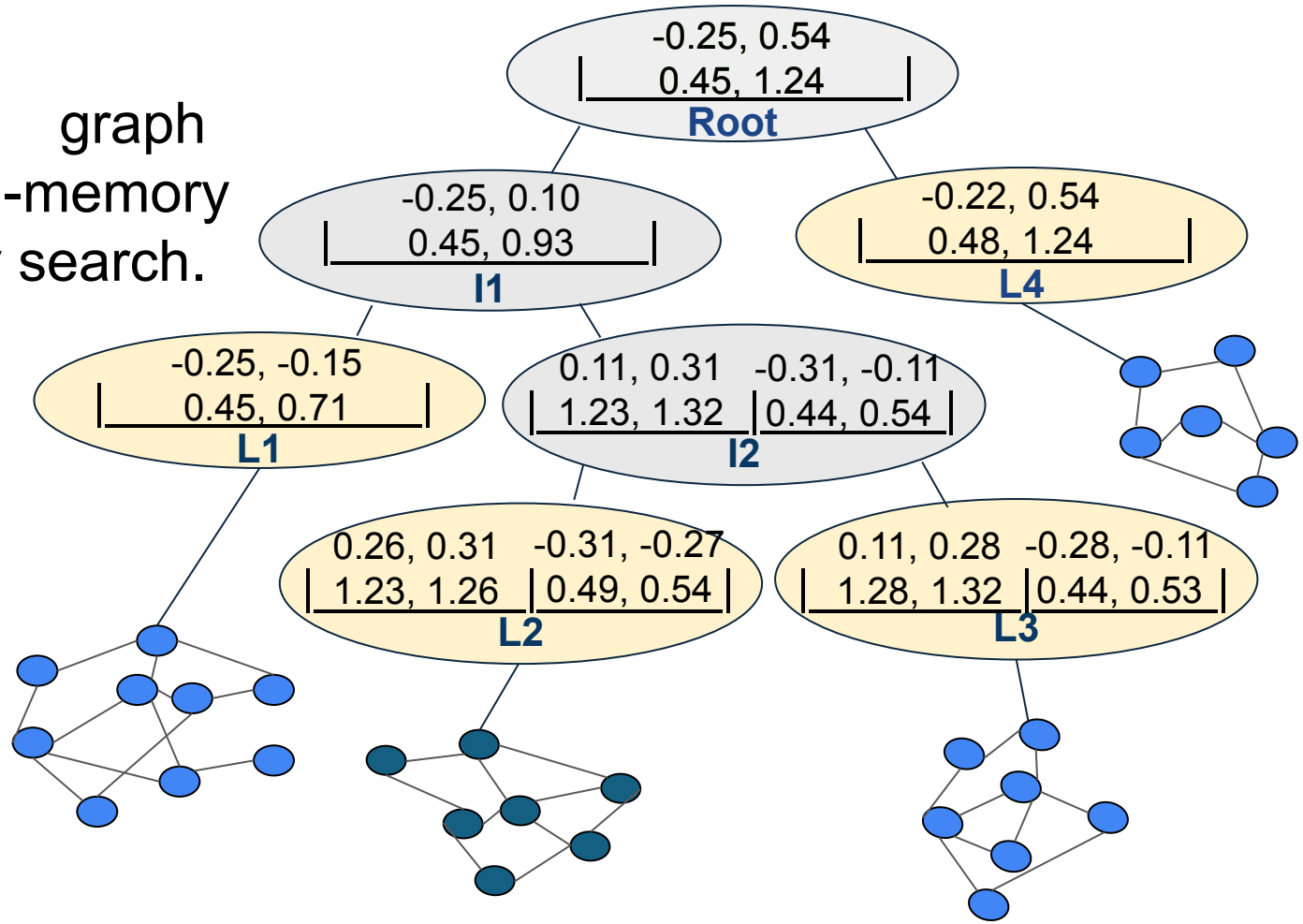
ELPIS

- **ELPIS** combines tree and graph structures for efficient in-memory ng-approximate vector similarity search.



Elpis Structure

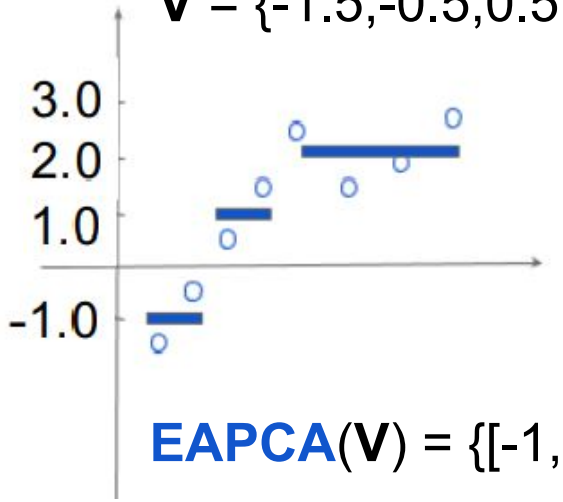
- **ELPIS** combines tree and graph structures for efficient in-memory ng-approximate vector similarity search.



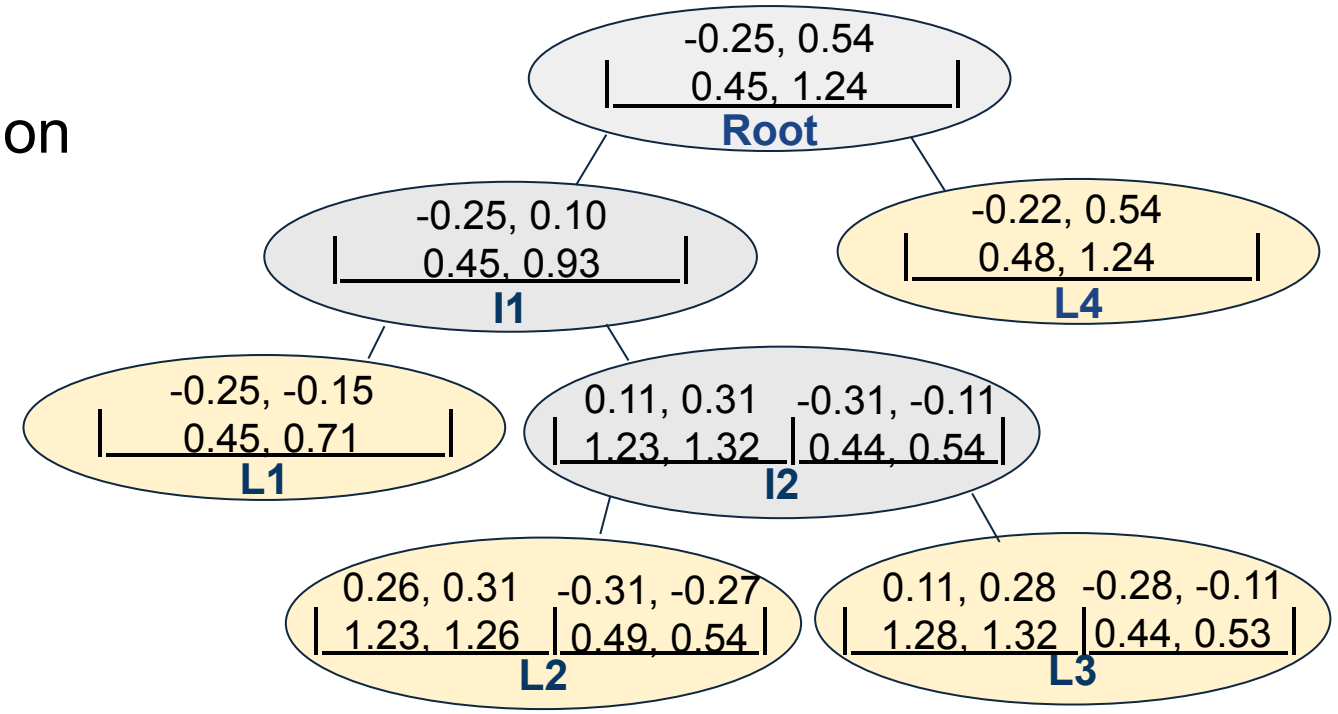
Elpis Index Building

- **EAPCA** data series summarization

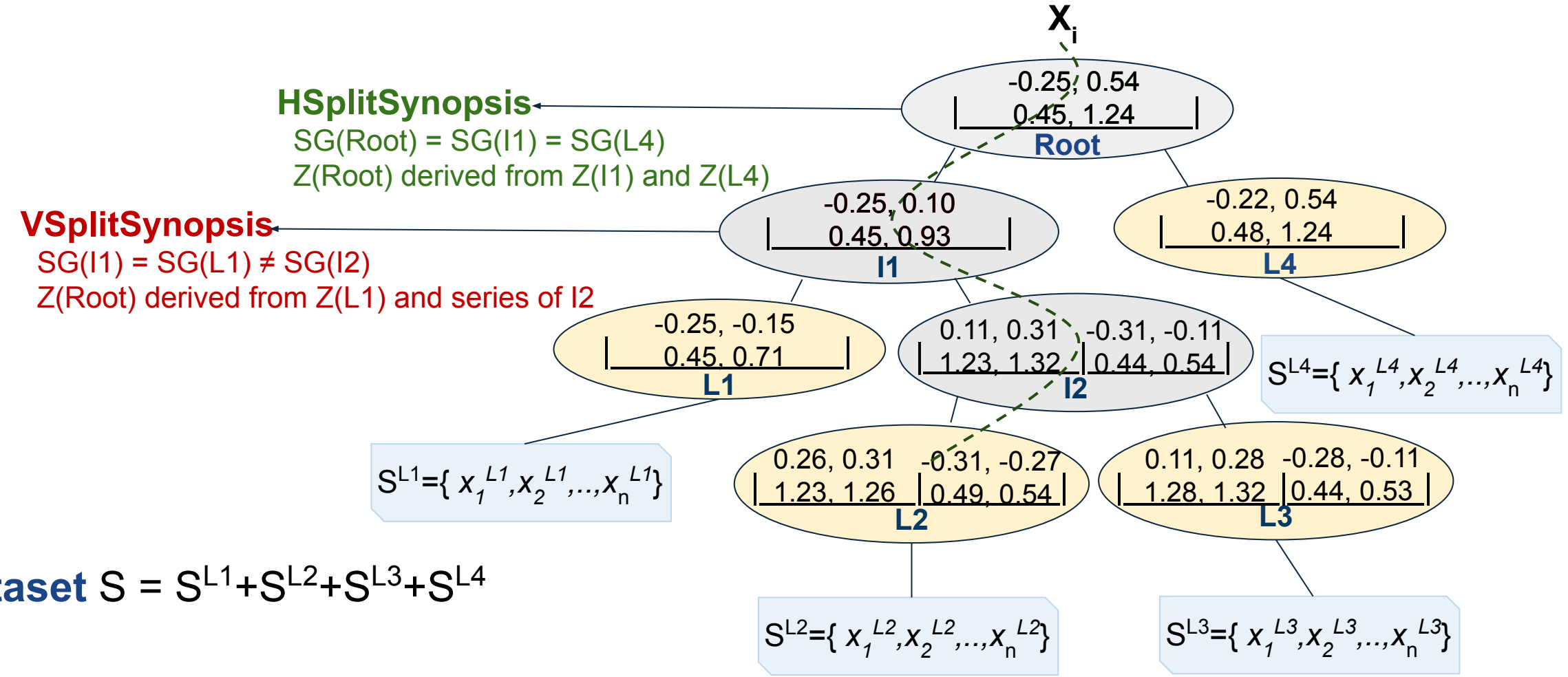
V = {-1.5,-0.5,0.5,1.5,2.5,1.5,2,2.6}



EAPCA(**V**) = {[-1,0.5],[1,0.5],[2.15,0.44]}



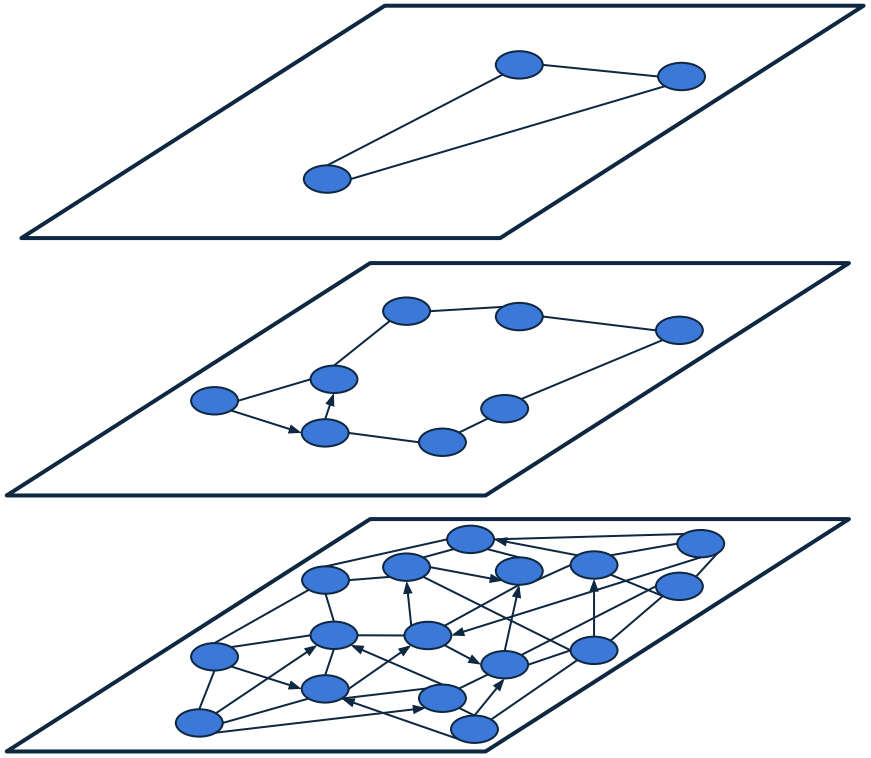
Elpis Index Building



- **Dataset** $S = S^{L1}+S^{L2}+S^{L3}+S^{L4}$

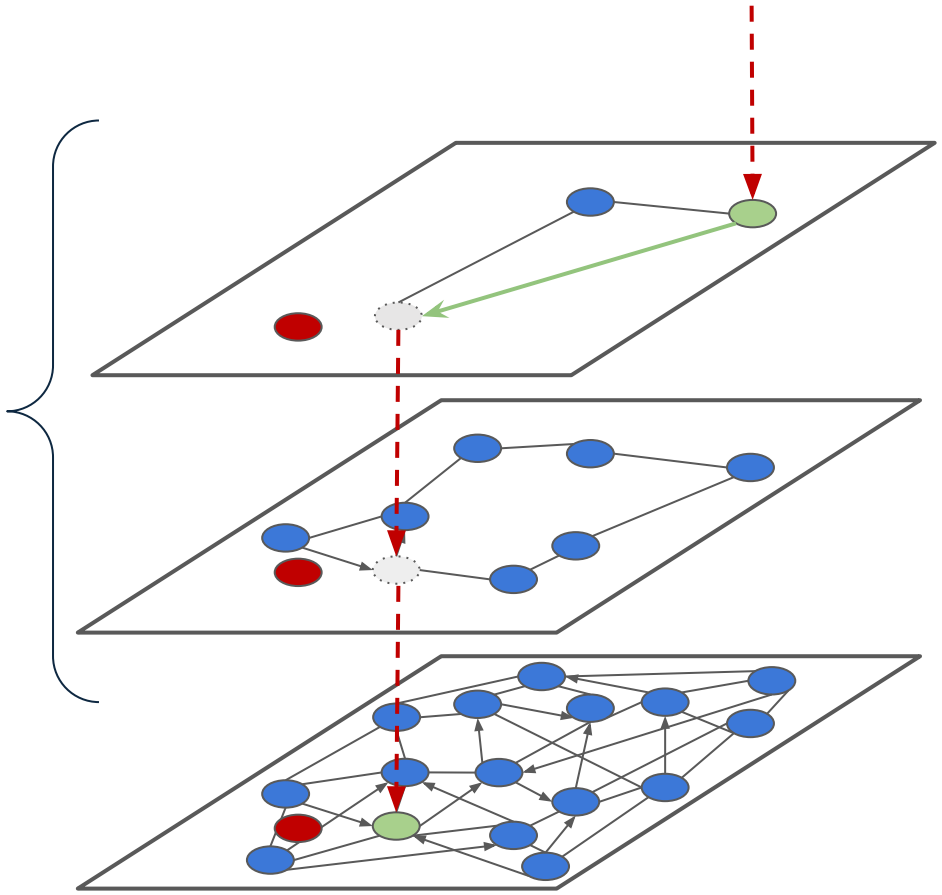
Graph Leaf: Hierarchical Navigable Small World

- HNSW stack multiple NSW into a mutli-resolution layers of approximate NSW graphs to solve connectivity problem

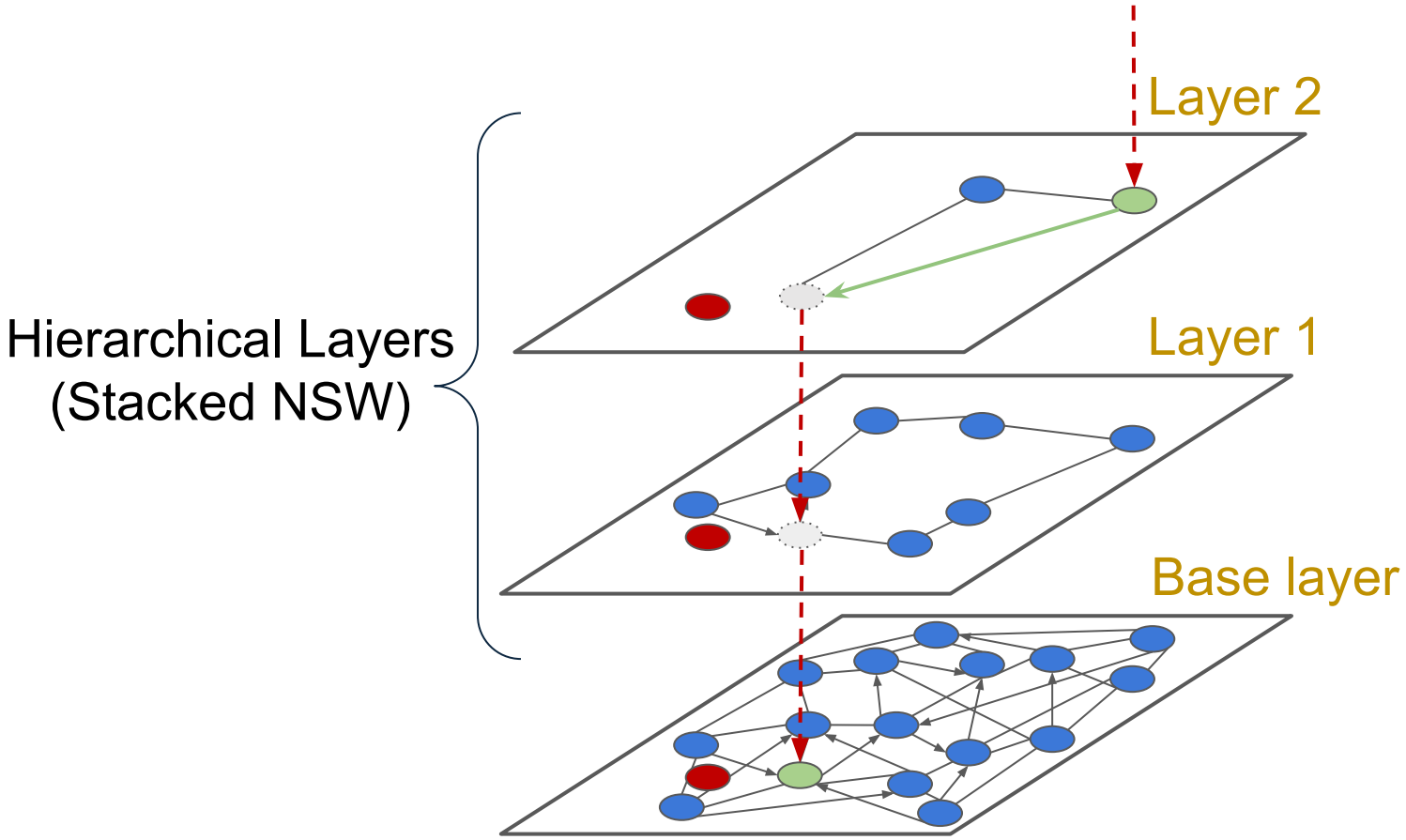


Graph Leaf: Hierarchical Navigable Small World

- HNSW stack multiple NSW into a mutli-resolution layers of approximate NSW graphs to solve connectivity problem
- HNSW's hierarchical layer edges act as long-range connections, aiding distance calculation pruning in searches

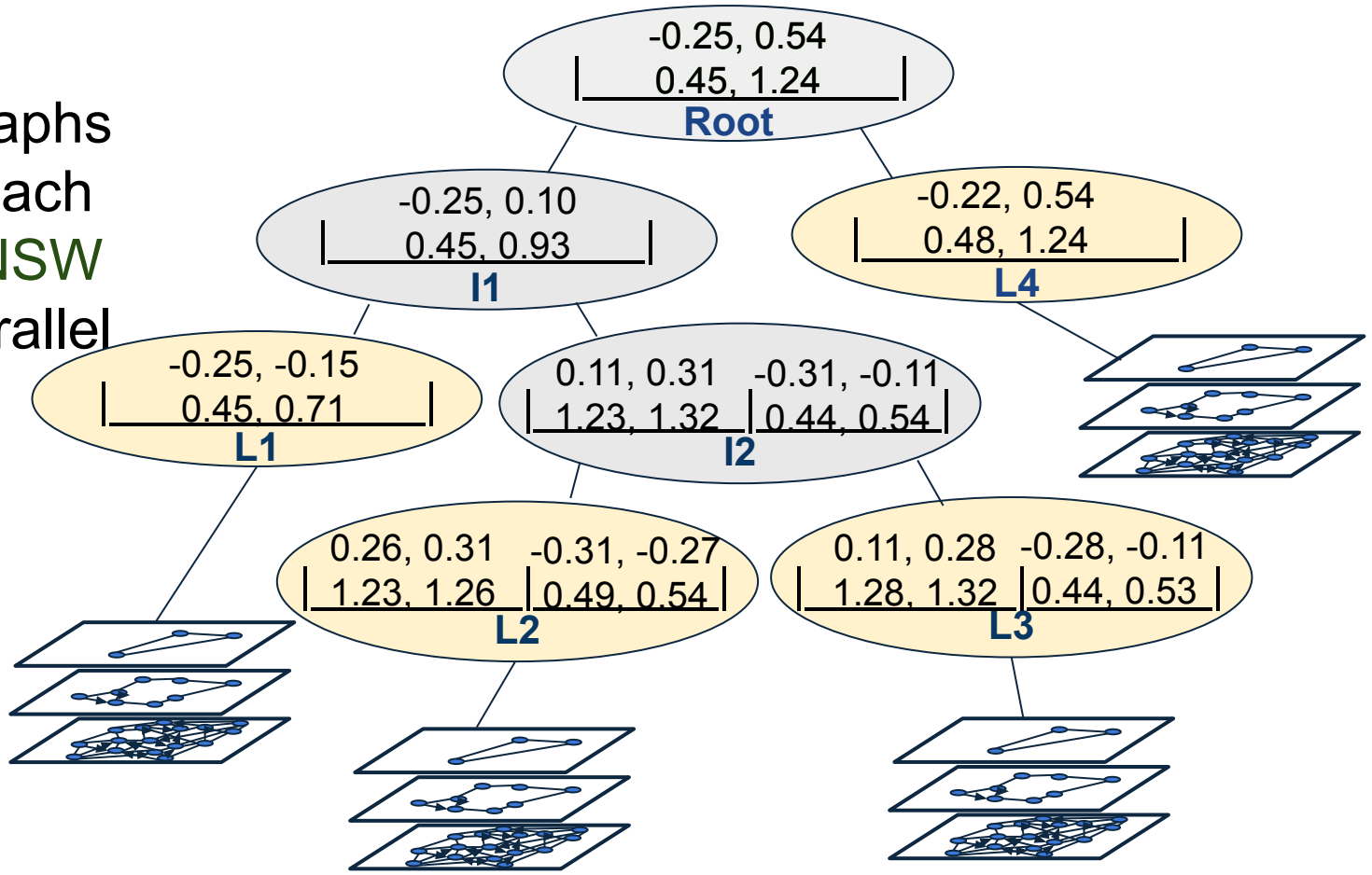


Graph Leaf: Hierarchical Navigable Small World



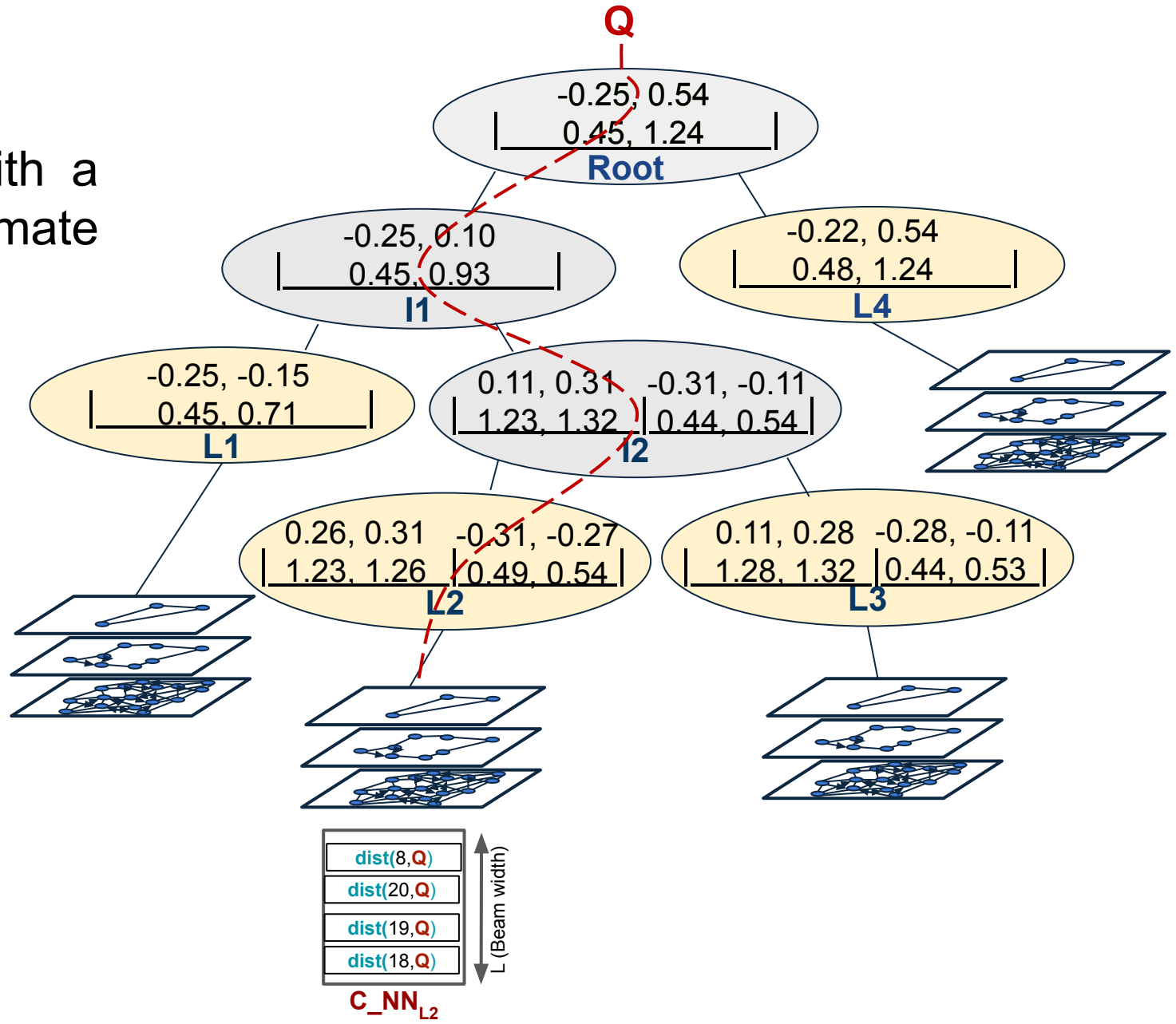
Elpis Index Building

- Elpis concurrently constructs graphs using *nw1* workers, with each worker building Hierarchical NSW graph within each leaf in parallel using *nw2* workers.



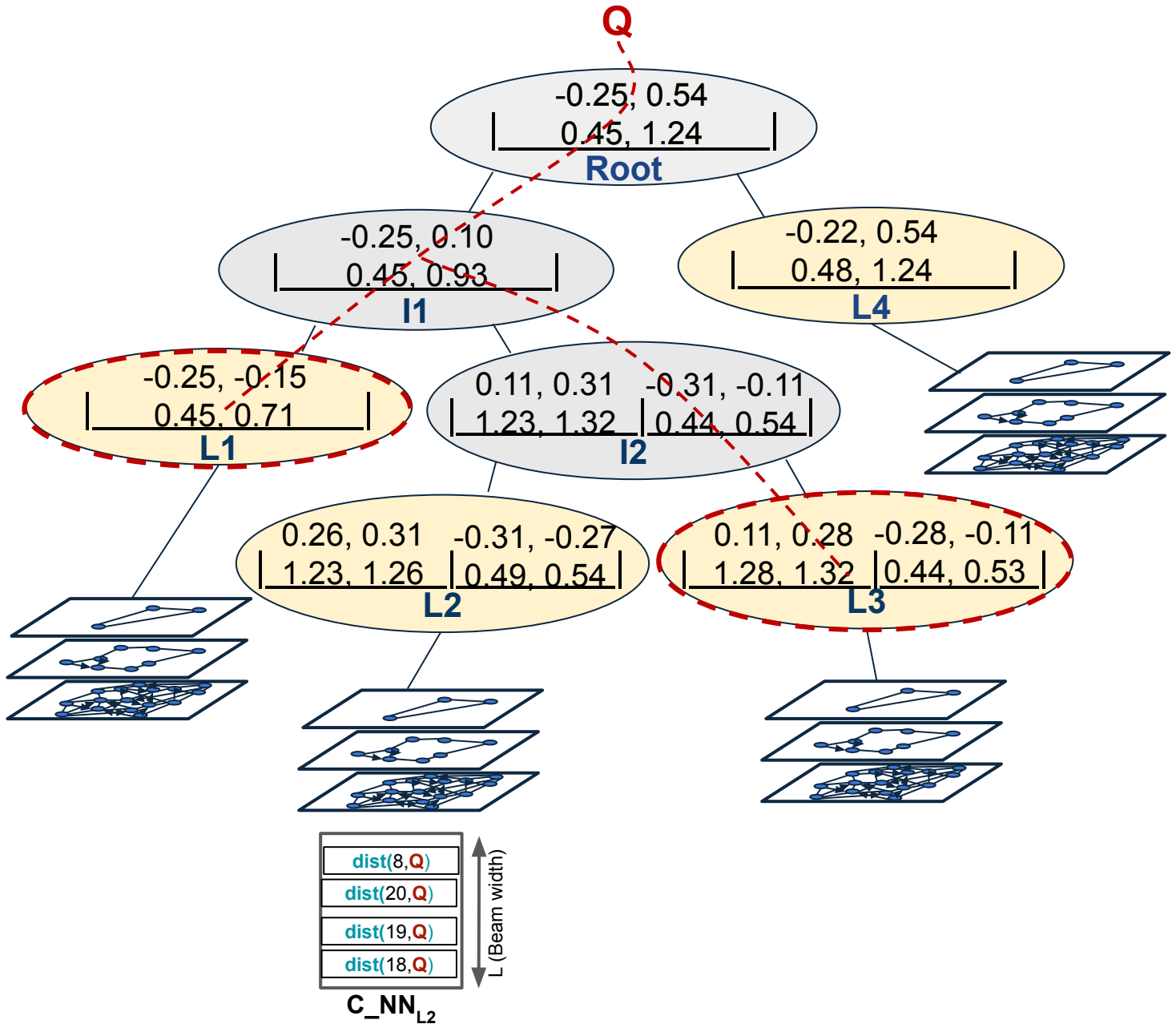
Elpis Query Search

- Elpis query search initiates with a beam search on the approximate graph leaf



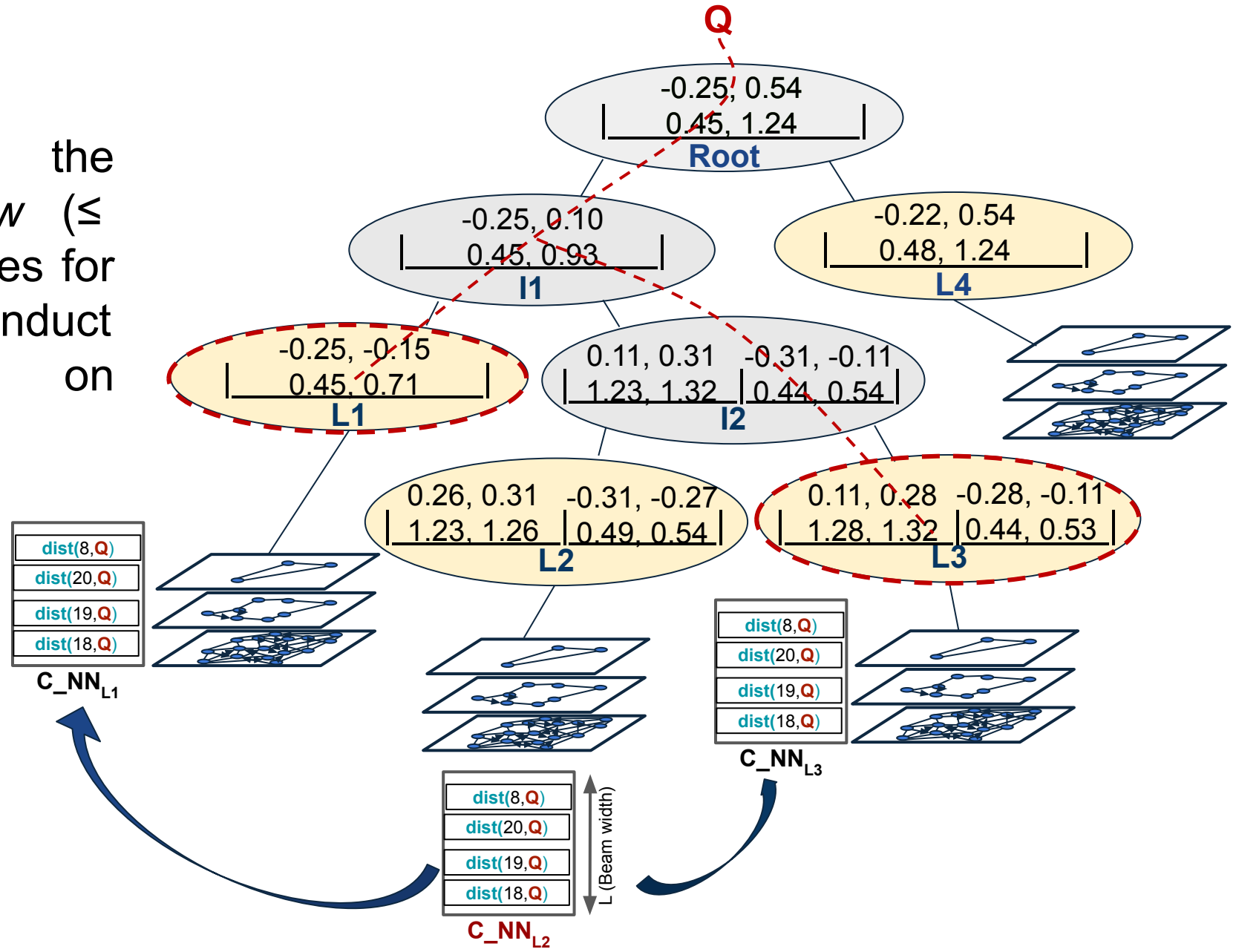
Elpis Query Search

- Elpis selects nprobes - 1 potential leaves through Breadth-First Search using K-bsf answers and EAPCA lower bounding distances.



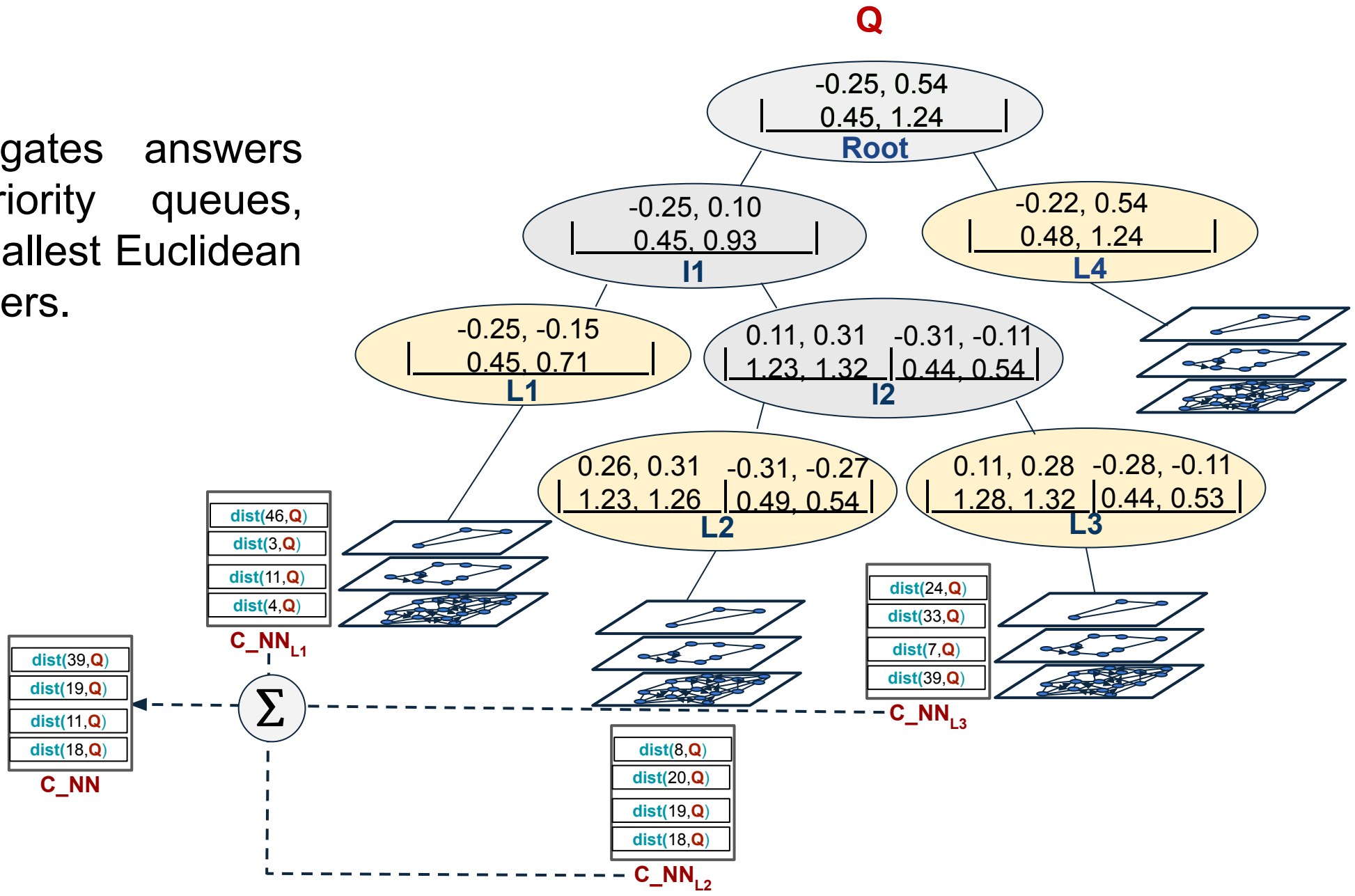
Elpis Query Search

- Elpis warms-up with the approximate results *nw* (≤ nprobes-1) priority queues for *nw* workers to conduct concurrent searches on nprobes - 1 graphs.

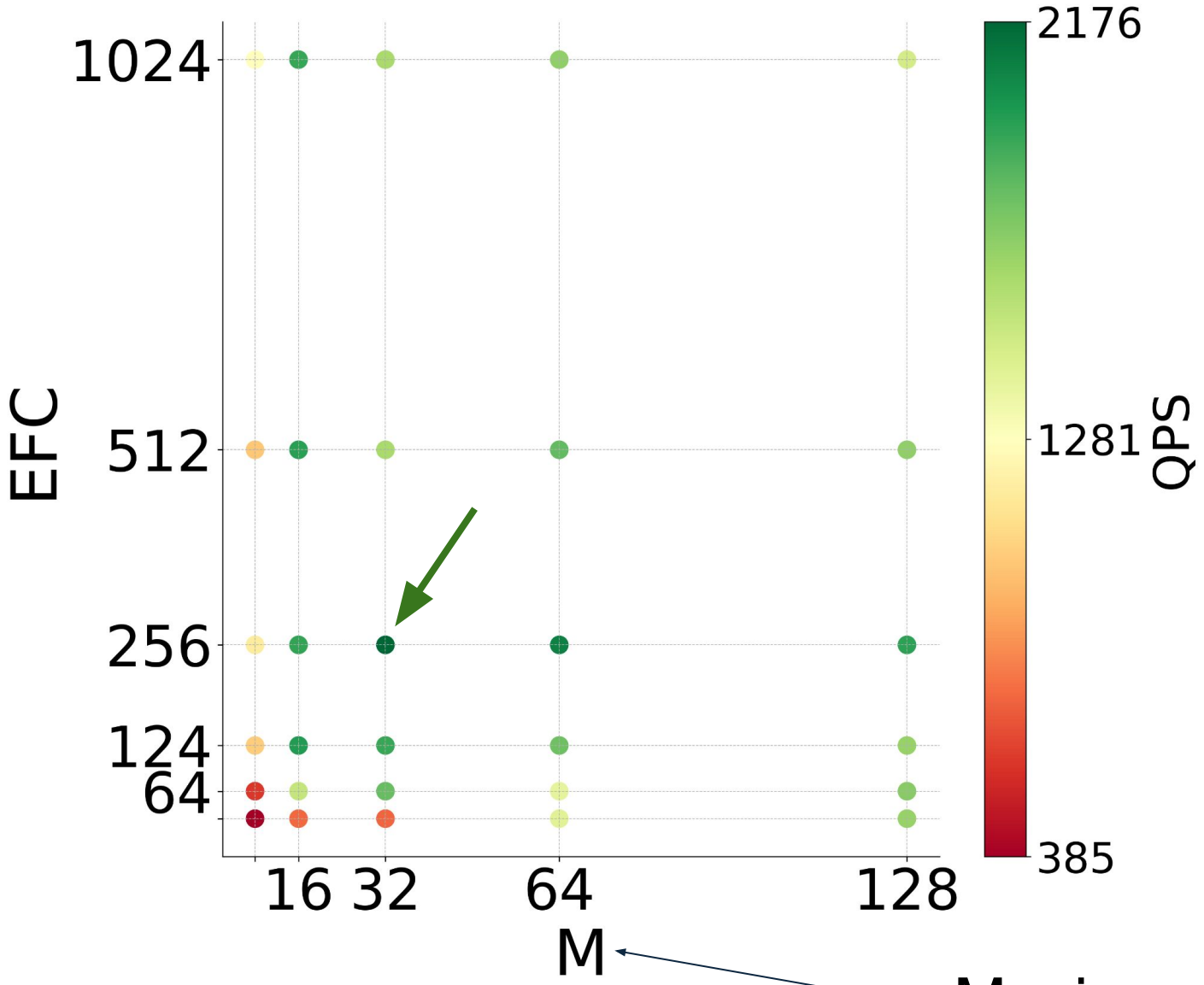


Elpis Query Search

- ELPIS aggregates answers from all priority queues, returning k smallest Euclidean distance answers.



HNSW Parameters (M, EFC)



Beam width

Maximum Outdegree