

MIRAGE: A FAST AND SCALABLE GRAPH-BASED INDEX FOR VECTOR SEARCH

M.TAMER ÖZSU & SAIRAJ VORUGANTI
UNIVERSITY OF WATERLOO

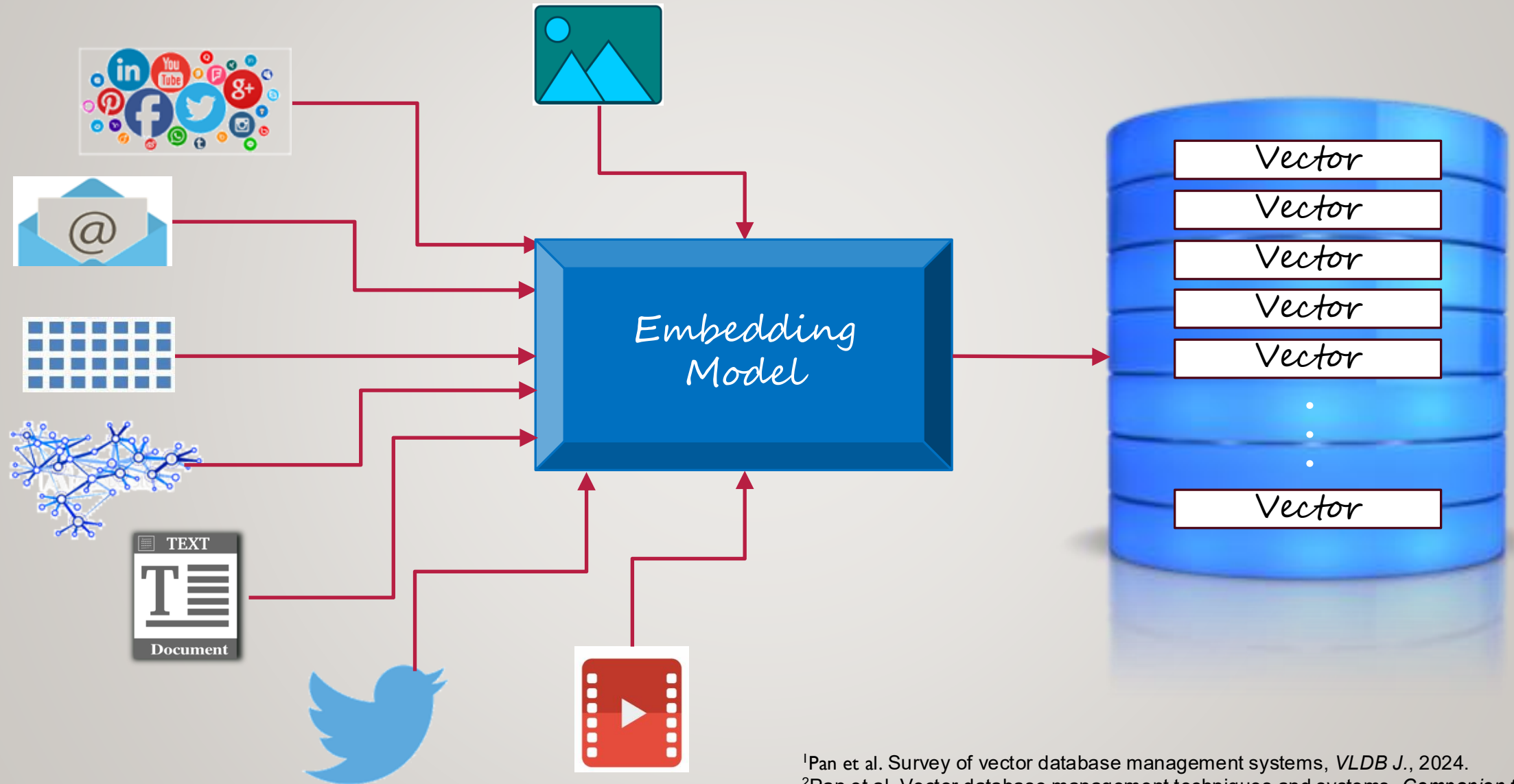
Proc. ACM Manag. Data, Vol. 3, No. 3 (SIGMOD), Article 188, June 2025



UNIVERSITY OF
WATERLOO



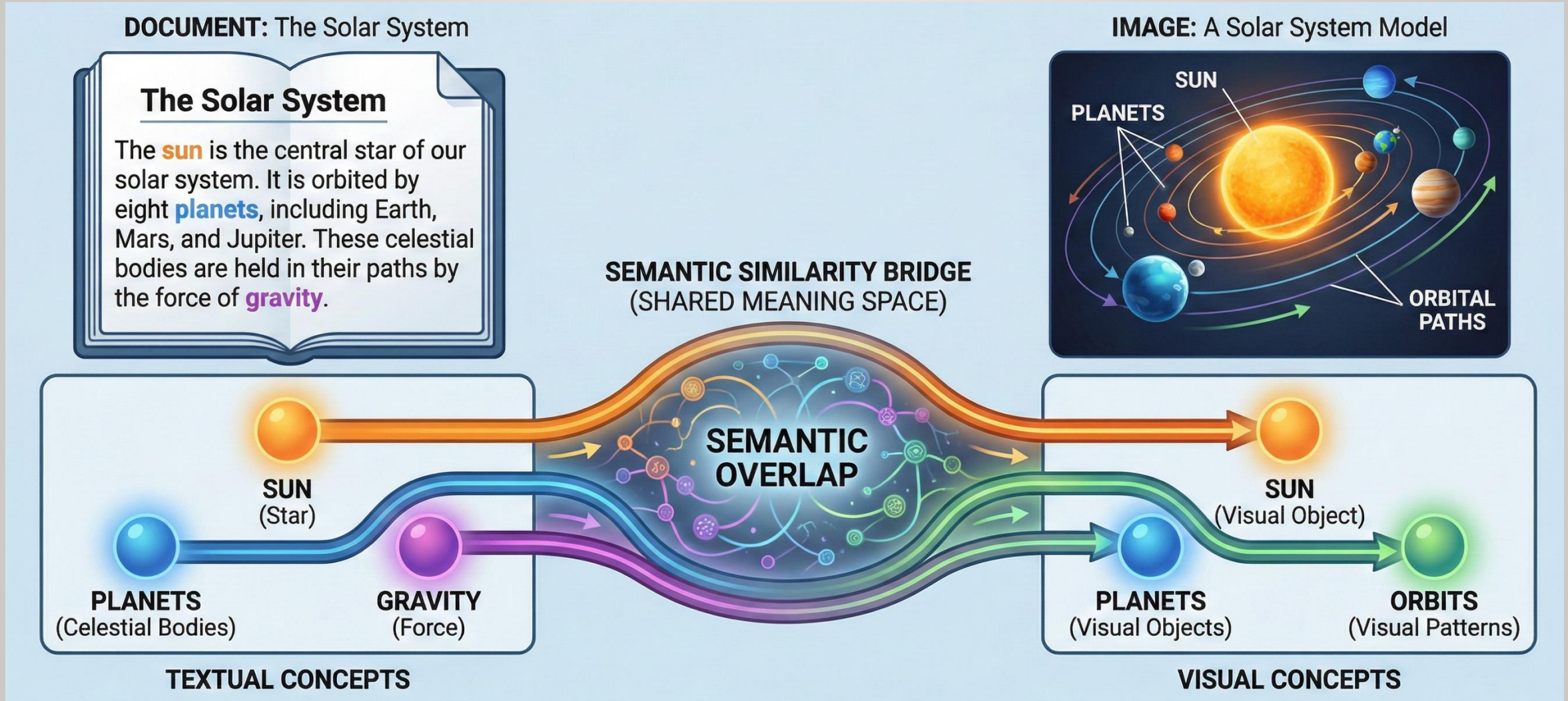
VECTOR DATABASES^{1,2}



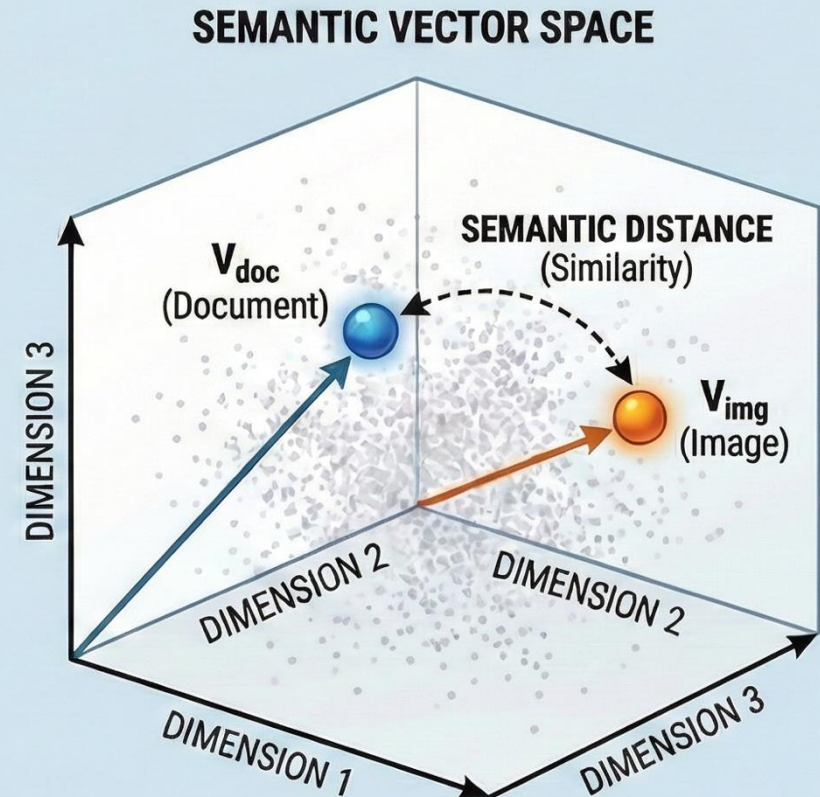
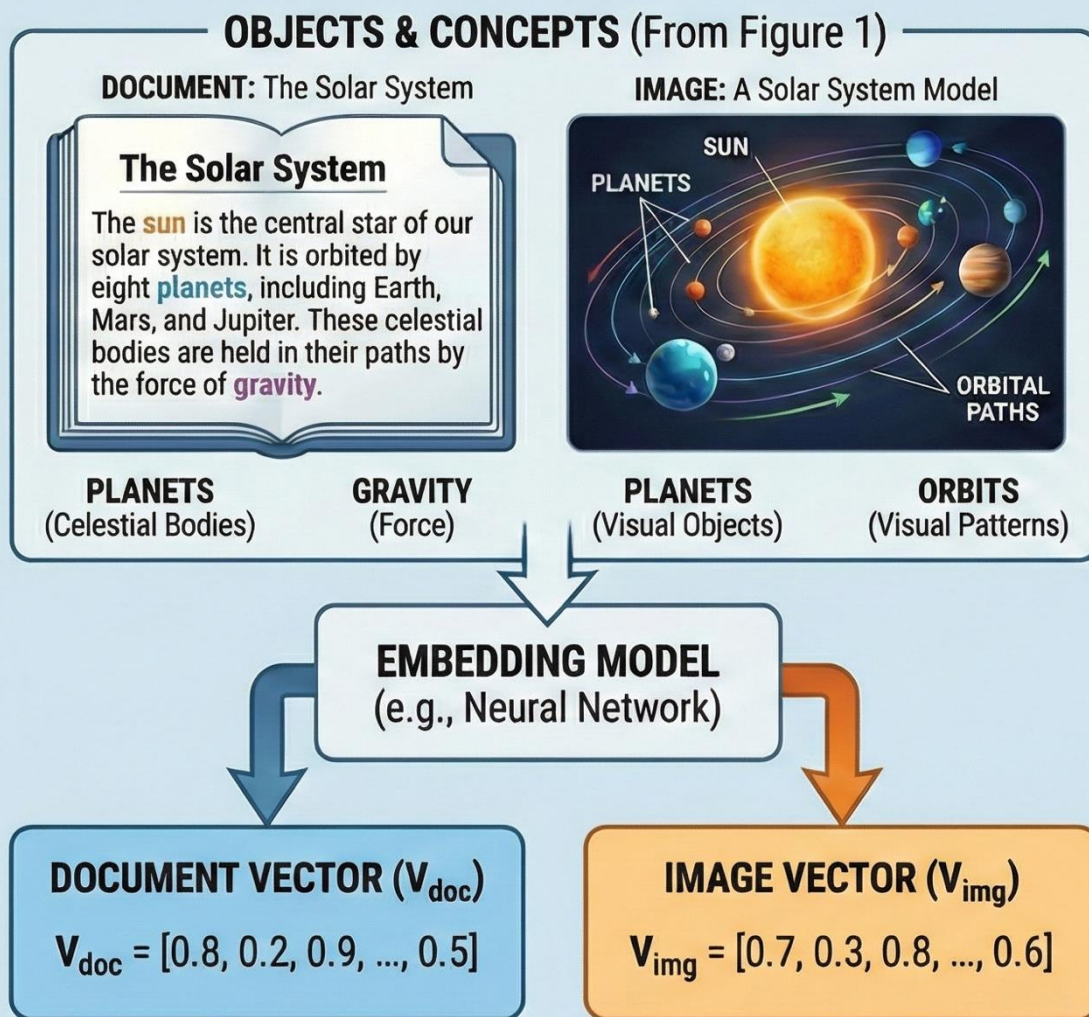
¹Pan et al. Survey of vector database management systems, *VLDB J.*, 2024.

²Pan et al. Vector database management techniques and systems. *Companion to SIGMOD*, 2024.

SEMANTIC SIMILARITY

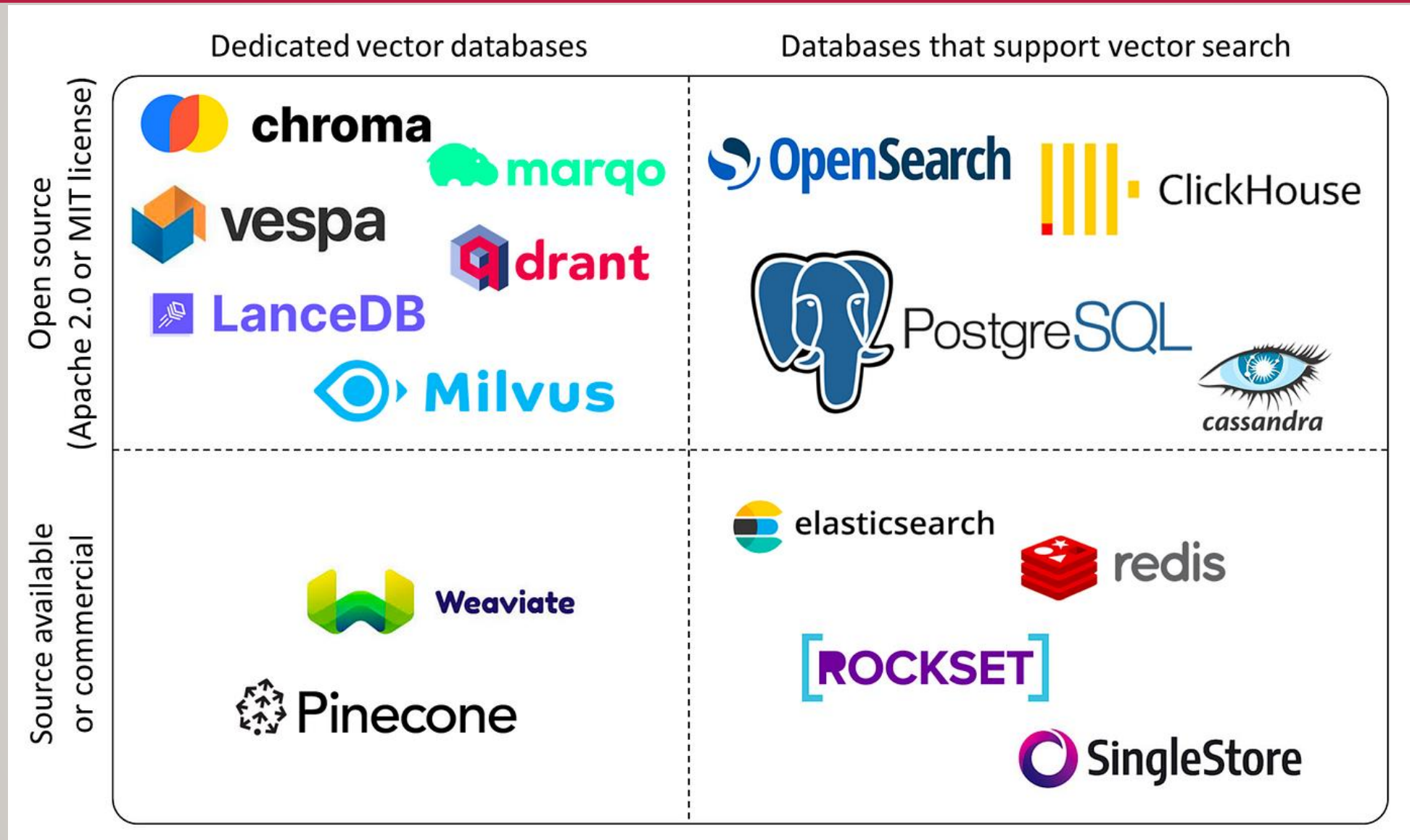


HOW VECTOR EMBEDDINGS HELP



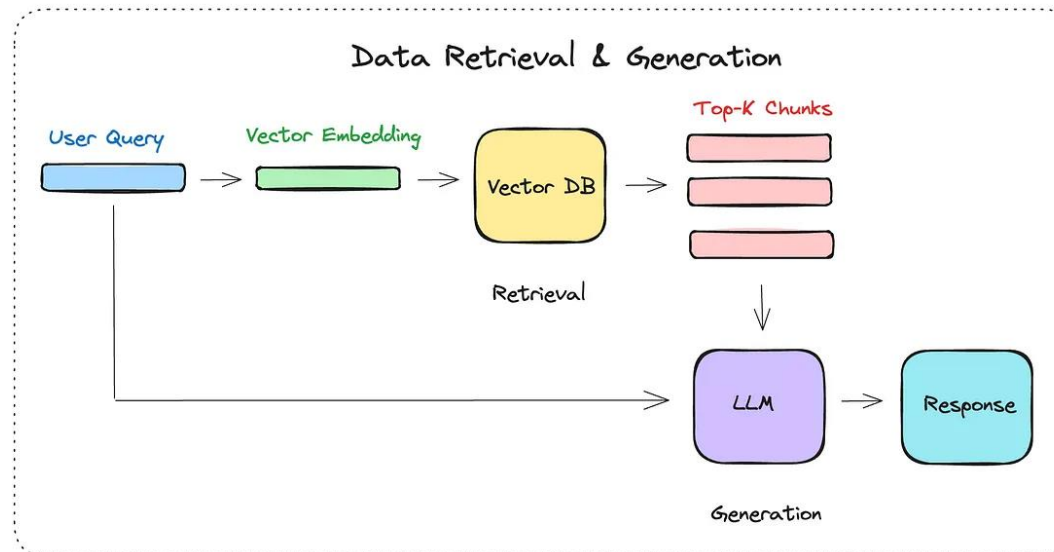
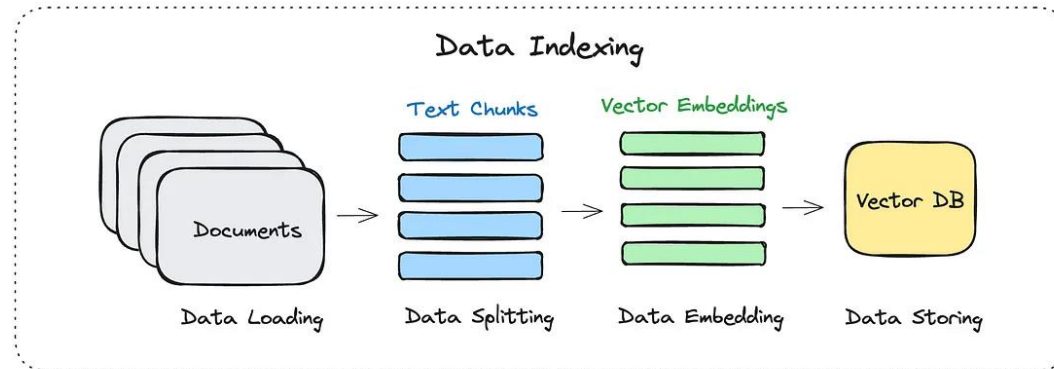
The embedding model maps the document and image to numerical vectors (V_{doc} , V_{img}) in a high-proximity enables efficient Approximate Nearest Neighbor (ANN) search to find related objects.

VECTOR DATA MANAGEMENT INDUSTRY



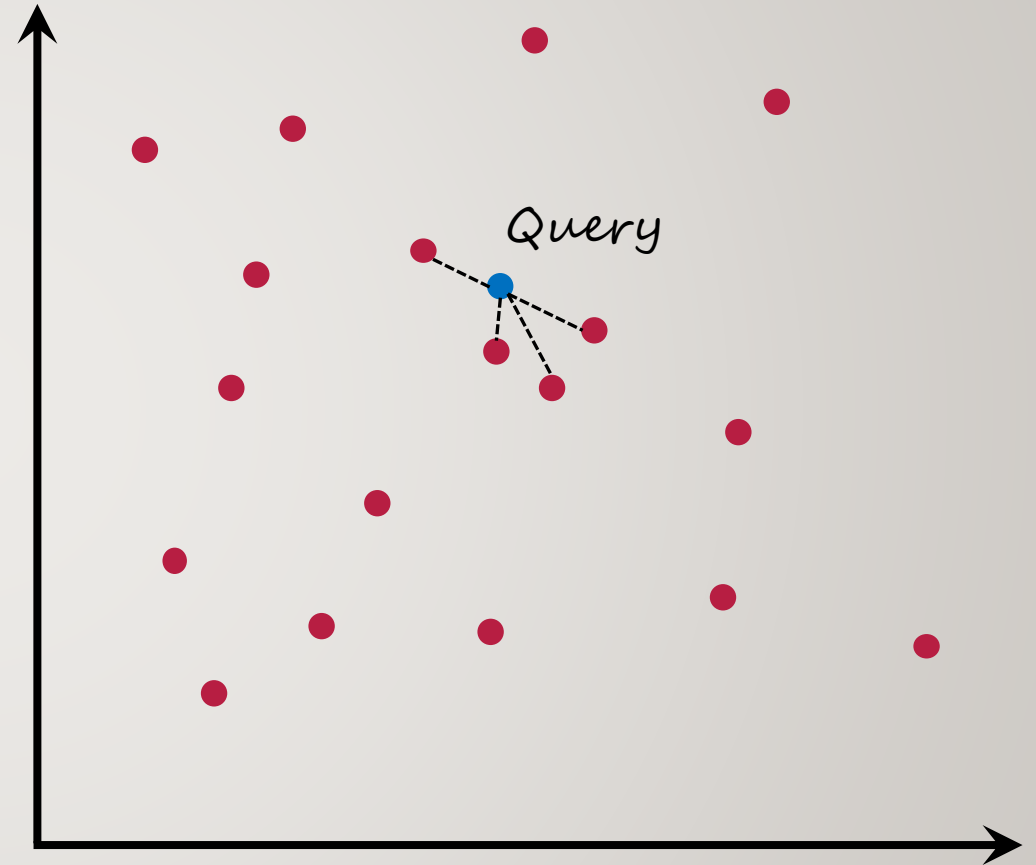
RETRIEVAL AUGMENTED GENERATION (RAG)

Basic RAG Pipeline



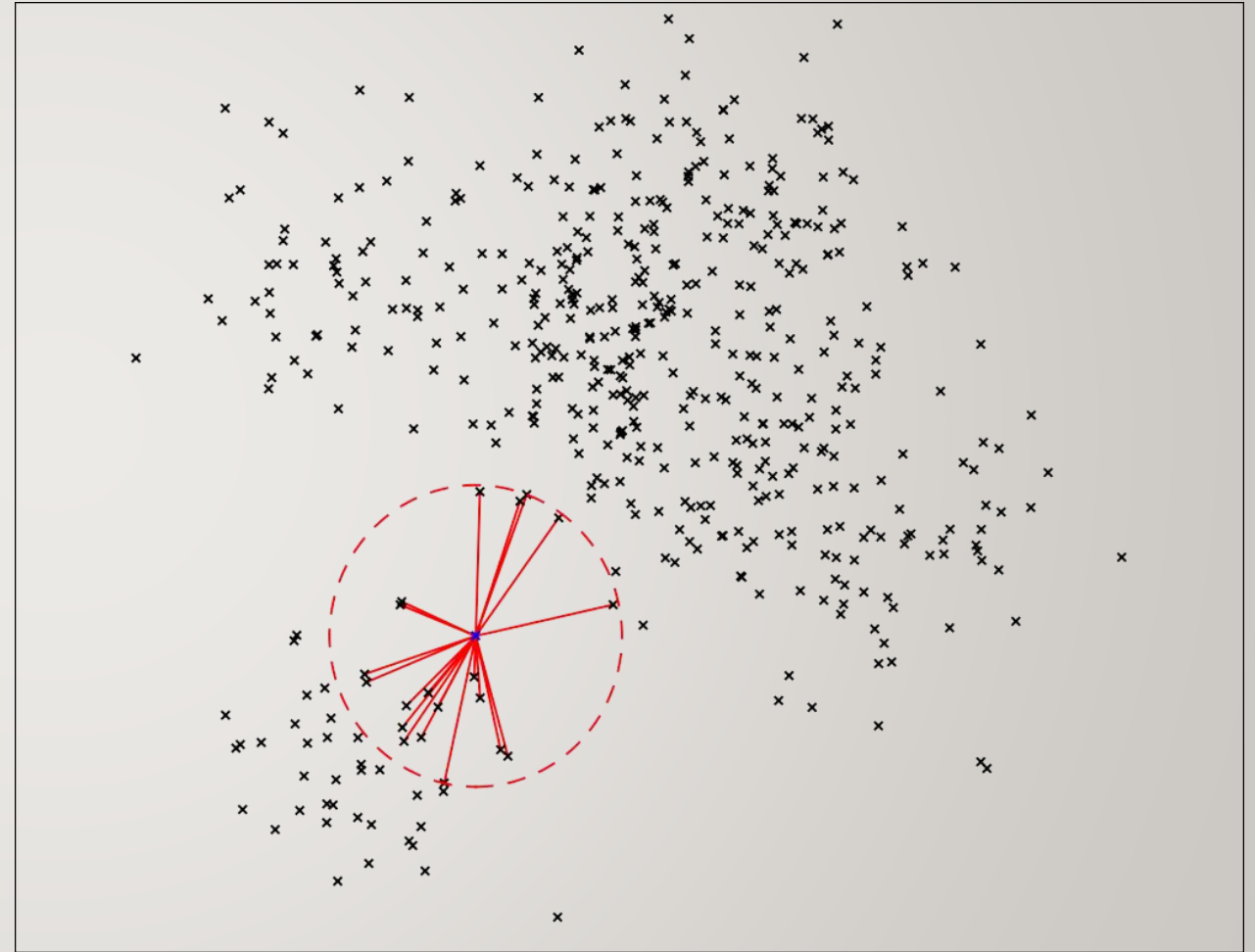
KNN SEARCH

- Retrieves the k closest vectors to the query point
- If the dimensionality becomes too high, need a linear scan to retrieve the nearest neighbors
 - Curse of dimensionality



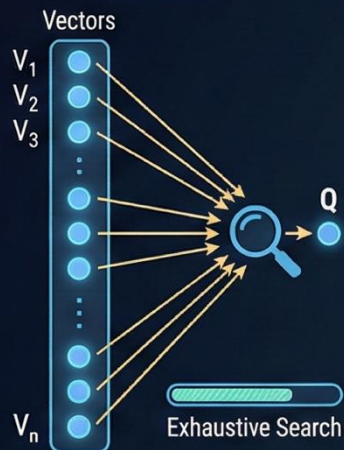
APPROXIMATE NEAREST NEIGHBOR SEARCH (ANNS)

- Reduces computational complexity
- Trade exact recall for huge performance boost



VECTOR INDEX TYPES

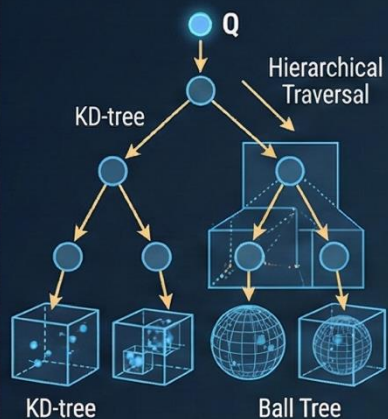
FLAT (BRUTE-FORCE) INDEXES



Stores all vectors, exhaustive distance computations. Exact results, poor scaling.

 Accuracy (High)
Latency (High, Poor Scaling)

TREE-BASED INDEXES



Hierarchical partition of vector space. Effective in low/moderate dimensions, performance degrades in high dimensions.

 Accuracy (Moderate/High)
Latency (Moderate)

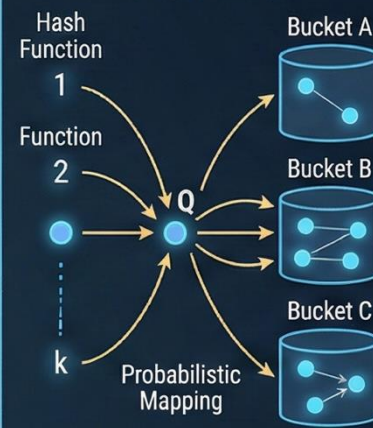
GRAPH-BASED INDEXES (e.g., HNSW)



Proximity graphs, greedy traversal. State-of-the-art ANN, high recall, high memory & construction cost.

 Accuracy (High, Approx.)
Latency (Low, Fast)
Memory (High)

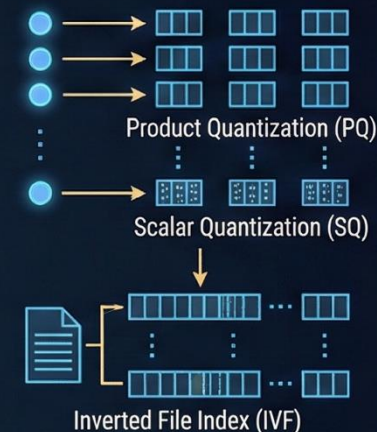
HASH-BASED INDEXES (e.g., LSH)



Randomized hashing, similar vectors in same buckets. Good scalability, lower recall than graphs.

 Accuracy (Moderate, Probabilistic)
Latency (Low)

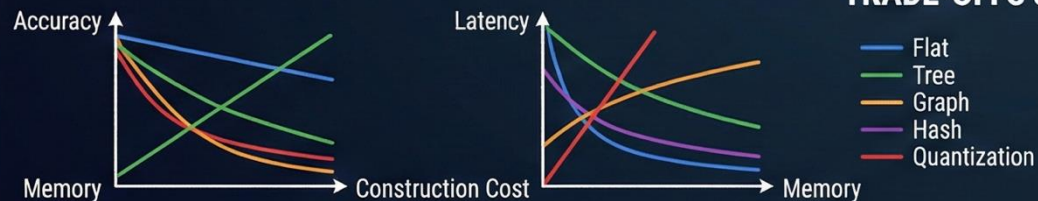
QUANTIZATION-BASED INDEXES



Compresses vectors into compact codes. Memory-efficient, cache-friendly distance computations. Often combined with IVF.

 Accuracy (Moderate/High, Approx.)
Latency (Low, Efficient)
Memory (Low, Efficient)

TRADE-OFFS & HYBRID INDEXES



IVF
+
Quantization

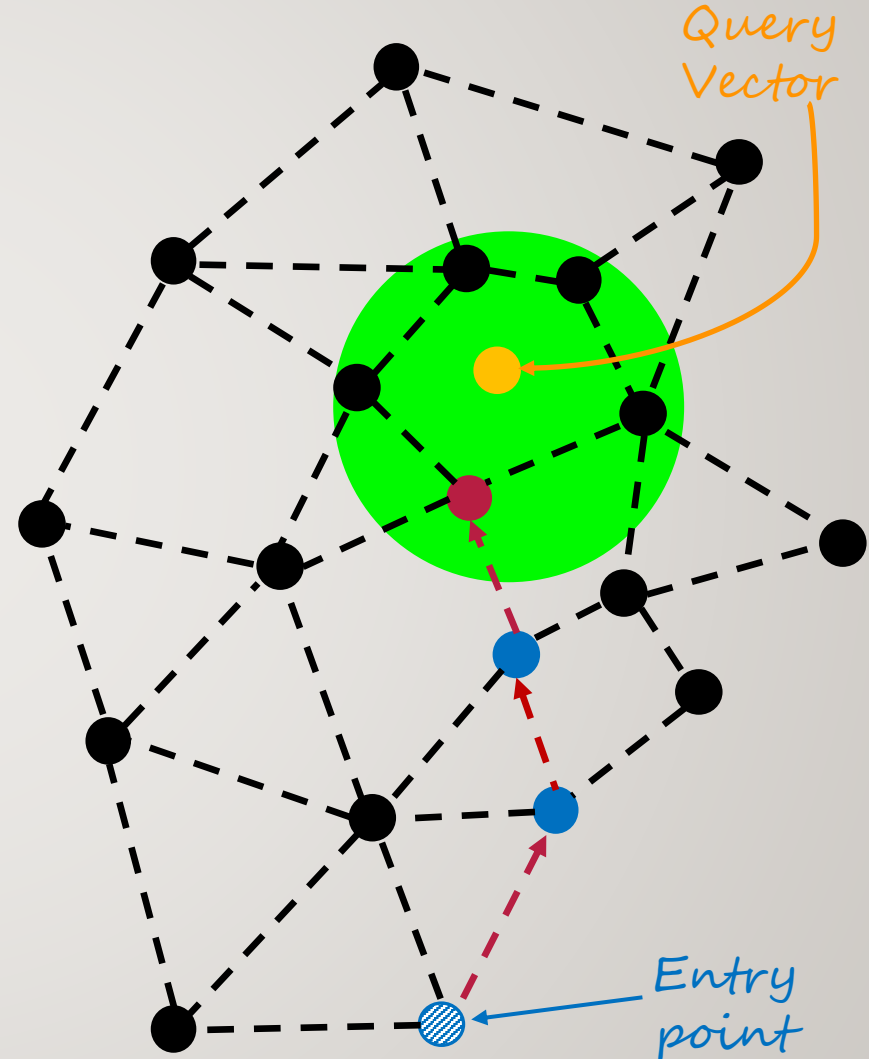
Graph
+
Compressed

HYBRID INDEX DESIGNS

Combines techniques (e.g., Inverted Indexes with Quantization) to balance accuracy, memory, and query latency at scale.

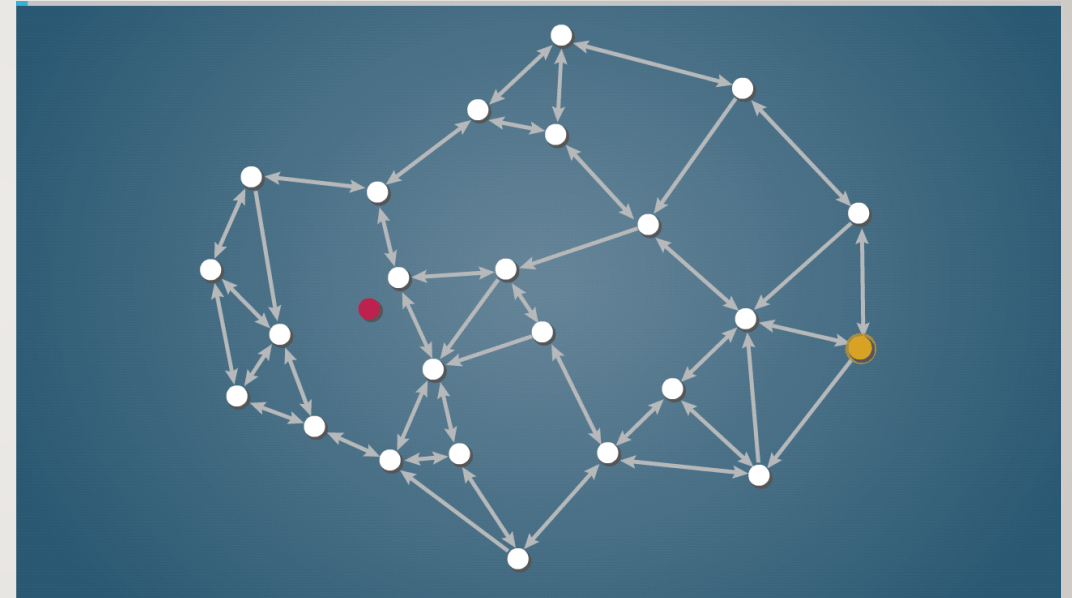
GRAPH-BASED INDEXES

- Vertices are vectors and edges connect similar neighbors
- Queries undergo a greedy search
- Pros: Search throughput and recall
- Cons: High construction time and memory cost



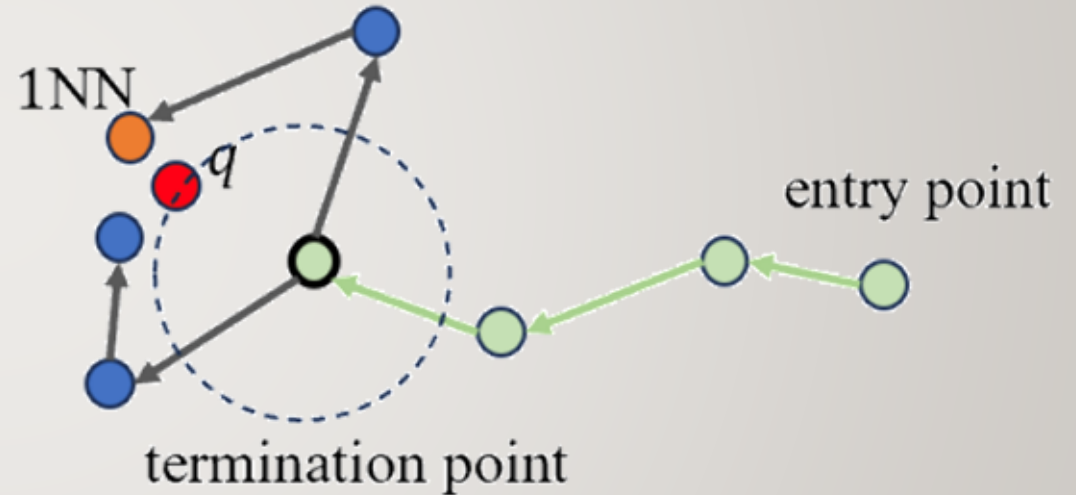
GRAPH-BASED INDEXES

- Vertices are vectors and edges connect similar neighbors
- Queries undergo a greedy search
- Pros: Search throughput and recall
- Cons: High construction time and memory cost



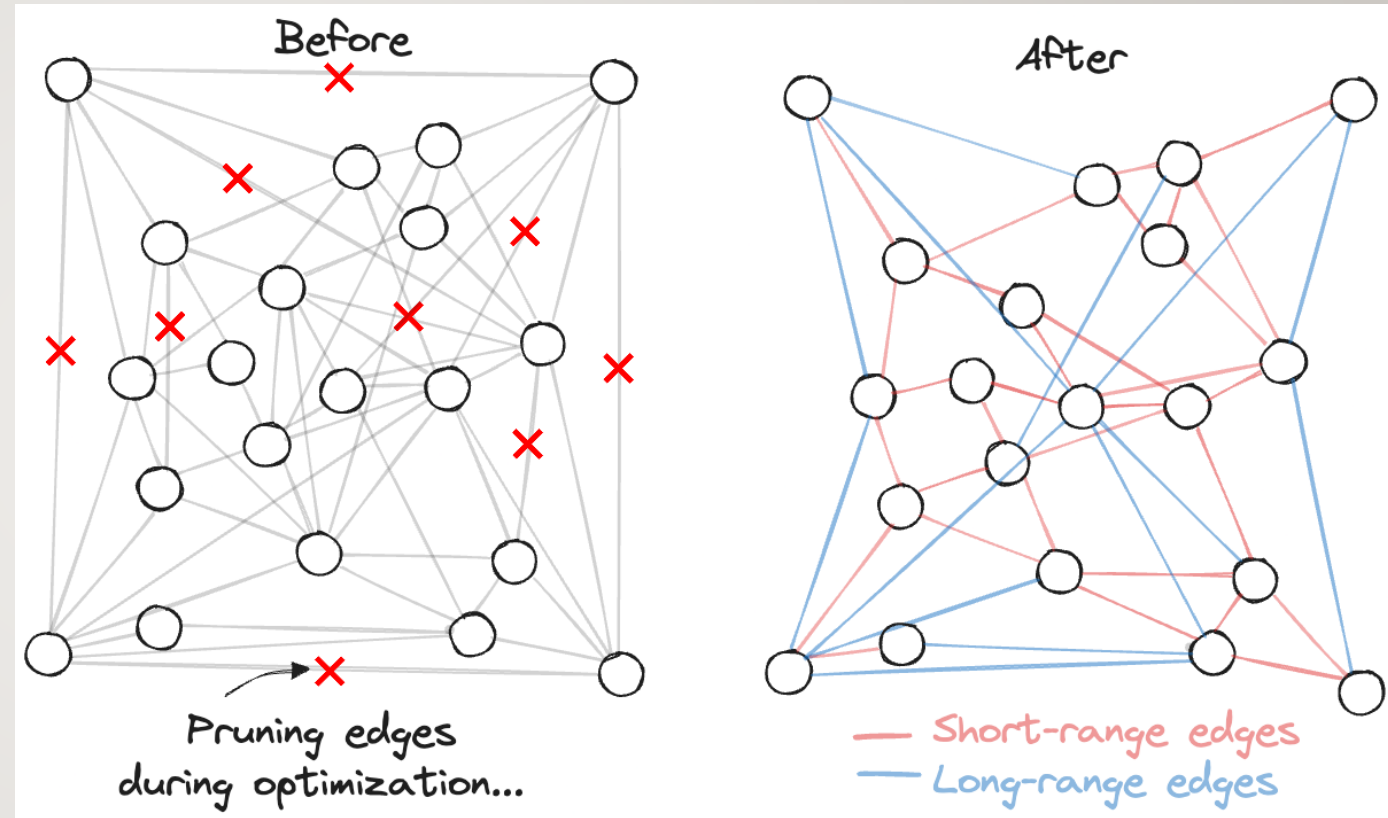
K-GRAPH

- Simplest graph-based index
 - Connect to k-nearest neighbors
- Issues
 - Vertices with high in-degrees (hubs) naturally connect to other hubs
 - Increases the likelihood of getting trapped in a locally optimal region



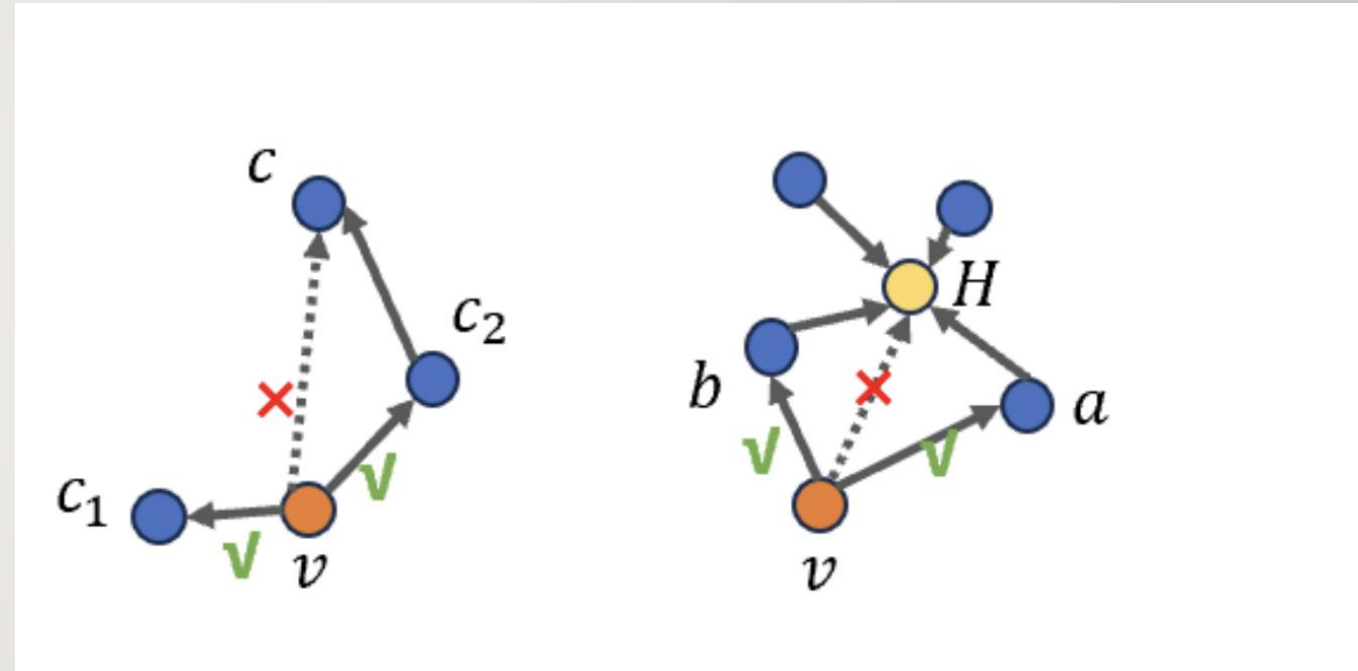
GRAPH-BASED INDEX INTUITION

- Want to have a good mix of long and short edges
- Don't want edges with too high of an out degree (hubs)



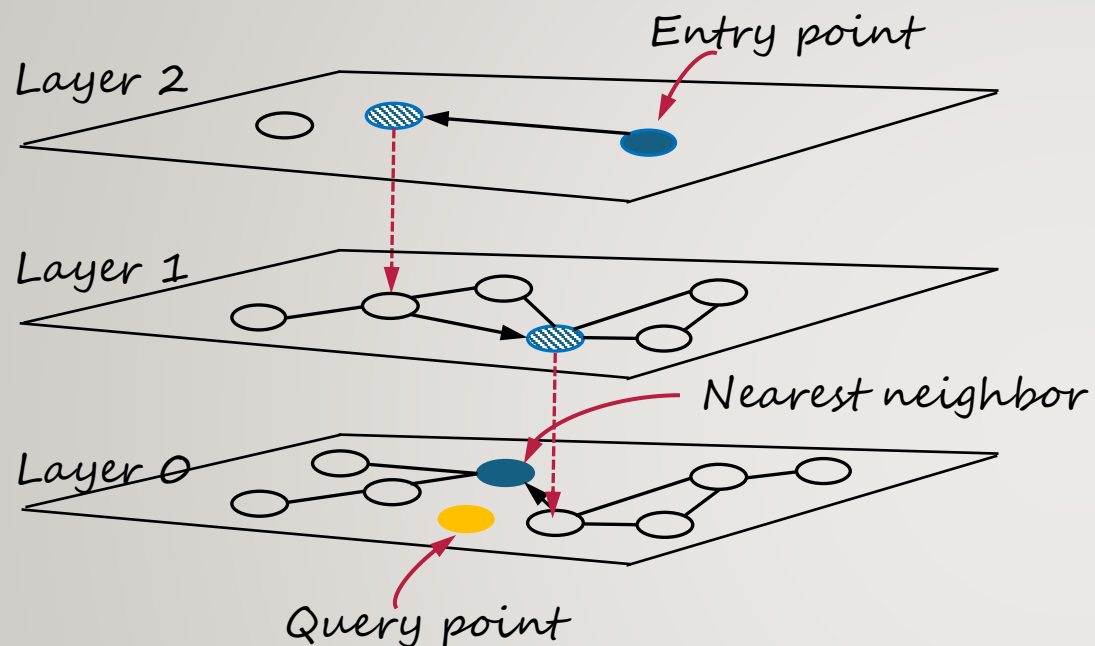
RNG RULE

- Prune the long edge of a triangle
- Keeps the navigability of the graph



GRAPH-BASED INDEX CONSTRUCTION

INCREMENT-BASED

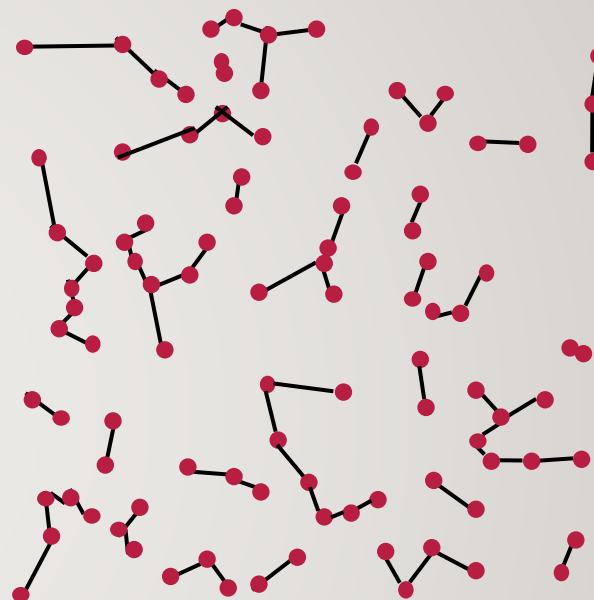


- E.g. Vamana¹, HNSW²
- Insert points one-by-one and treat each point as a query

¹ Subramanya et al. DiskANN: fast accurate billion-point nearest neighbor search on a single node. In Proc. 33rd Annual Conf. on Neural Information Processing Syst., 2019.

² Malkov and Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

REFINEMENT-BASED



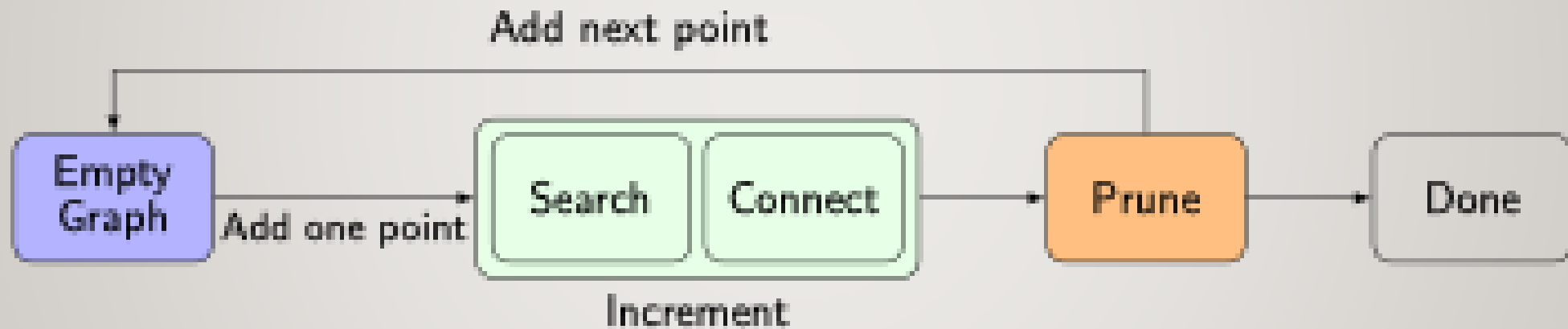
- E.g. KGraph³ (NNDescent⁴), NSG⁵
- Initialize random kNNG
- For each node, change any 2-hop neighbor into a 1-hop neighbor if it is a new k -nearest neighbor
- Repeat until convergence

³ Dong et al. Efficient k-nearest neighbor graph construction for generic similarity measures. *Proc. 20th Intl. Conf. on World Wide Web.*, 2011.

⁴ Bratić et al., NN-Descent on High-Dimensional Data. *Proc. 8th Intl. Conf. on Web Intelligence, Mining and Semantics*, 2018.

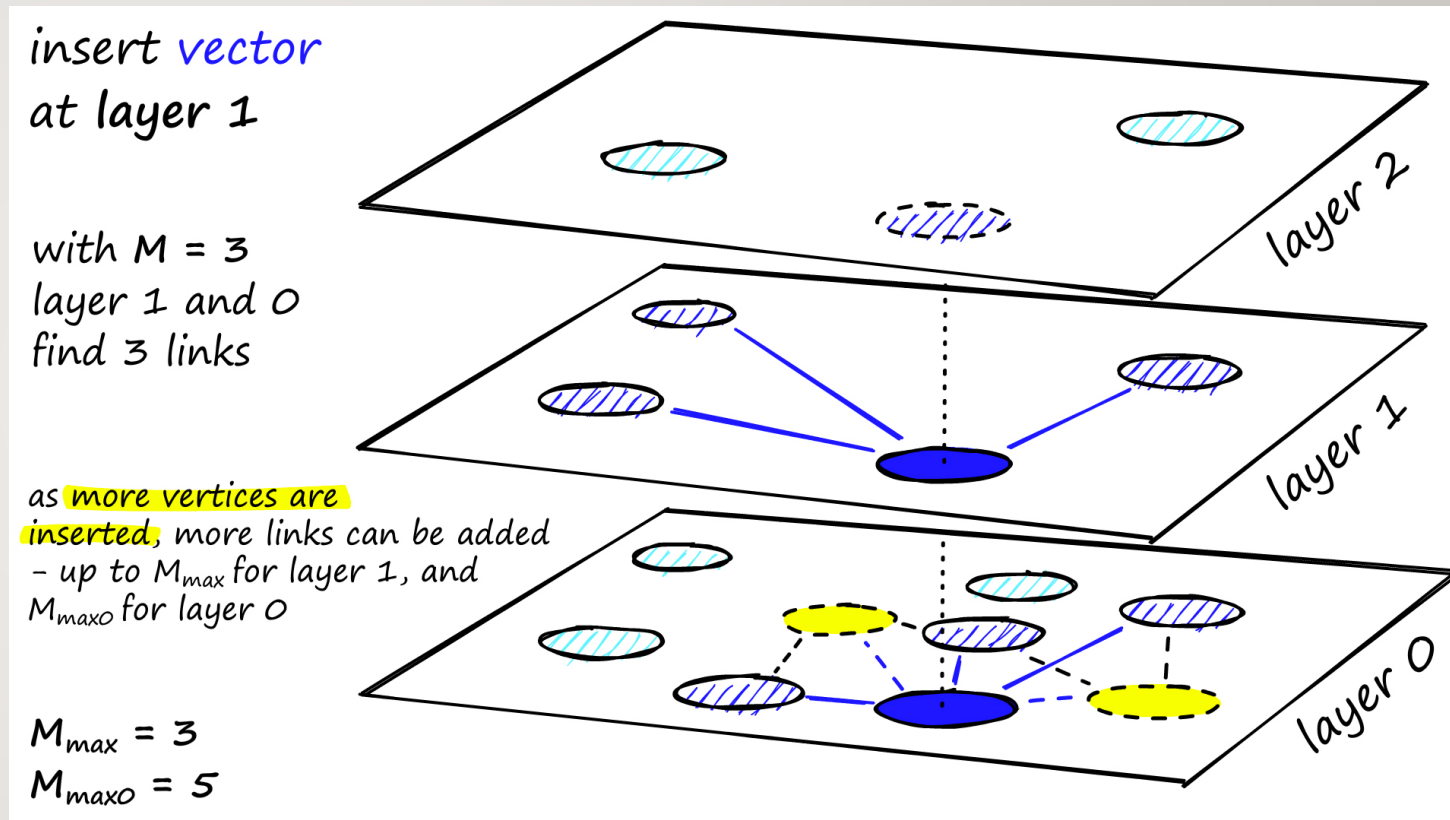
⁵ Fu et al. Fast approximate nearest neighbor search with the navigating spreading-out graph. *PVLDB*, 2019.

INCREMENT-BASED CONSTRUCTION



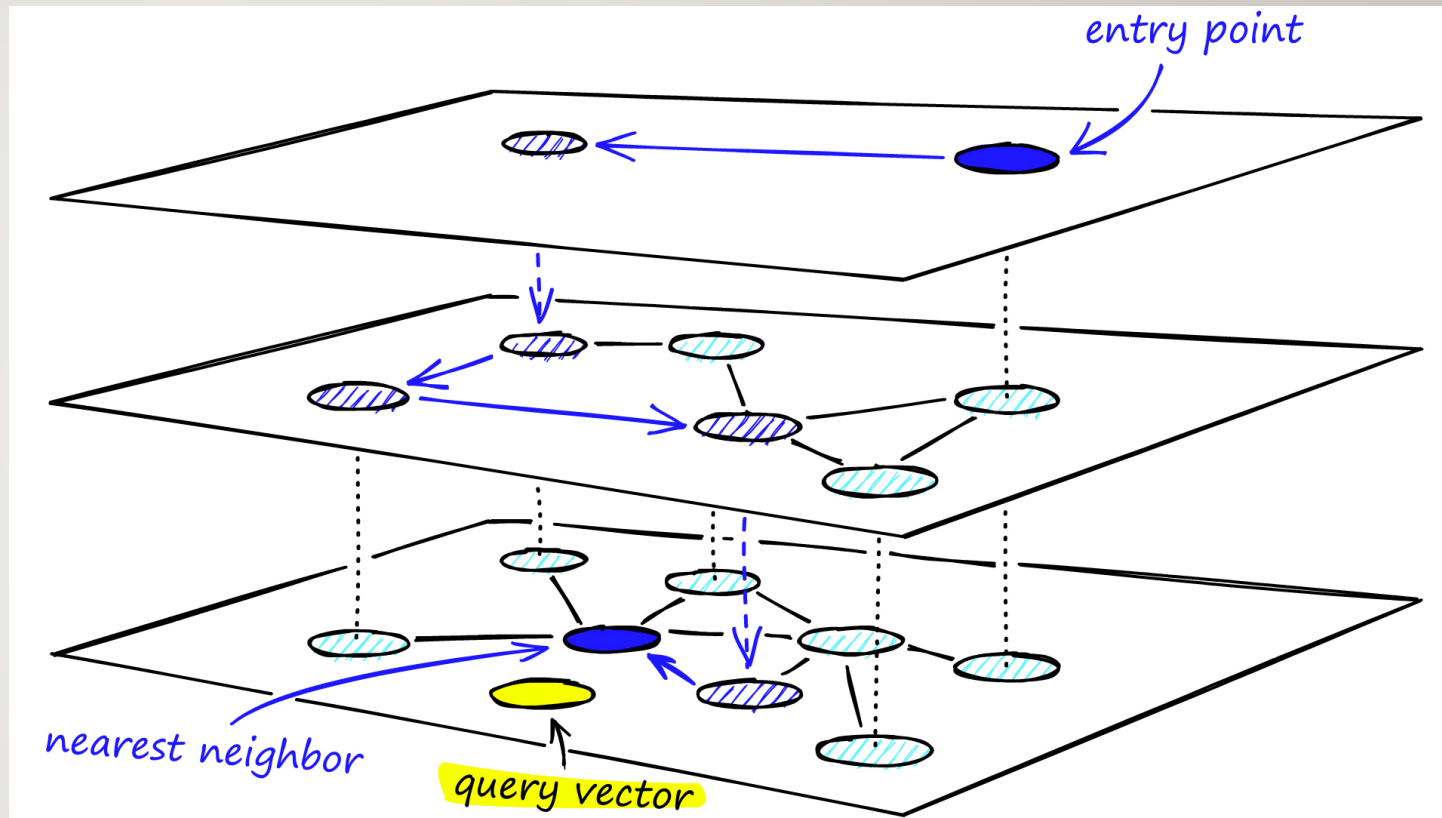
INCREMENT-BASED CONSTRUCTION (HNSW)

- Inserts happen one at a time
- Each point treated as a query
 - Performs ANNS to find neighbors
- Probabilistically decide which layer the vector will be inserted into
 - A new vector will appear in upper layers with an exponentially decreasing probability
 - Layer 0 contains all vectors

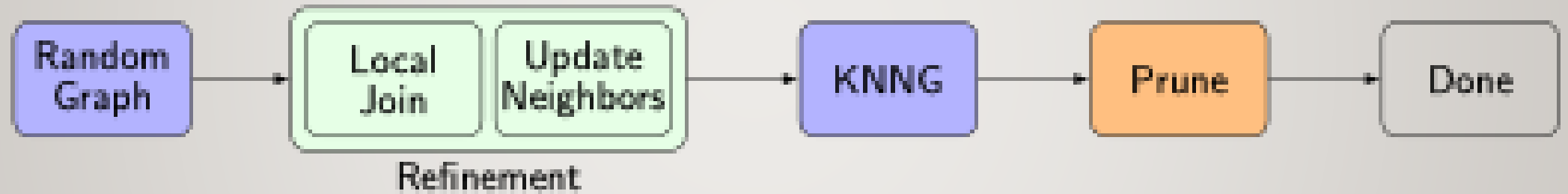


HNSW SEARCH

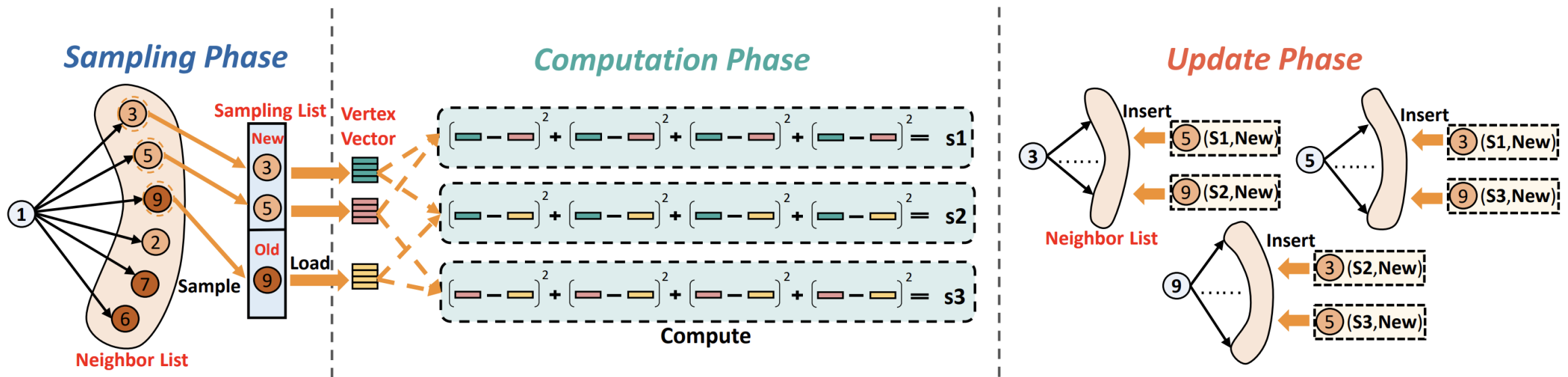
- Fixed entry point at top layer
- Find the nearest neighbor to the query point at each layer, move down to the next layer, and continue through layer 0



REFINEMENT-BASED CONSTRUCTION

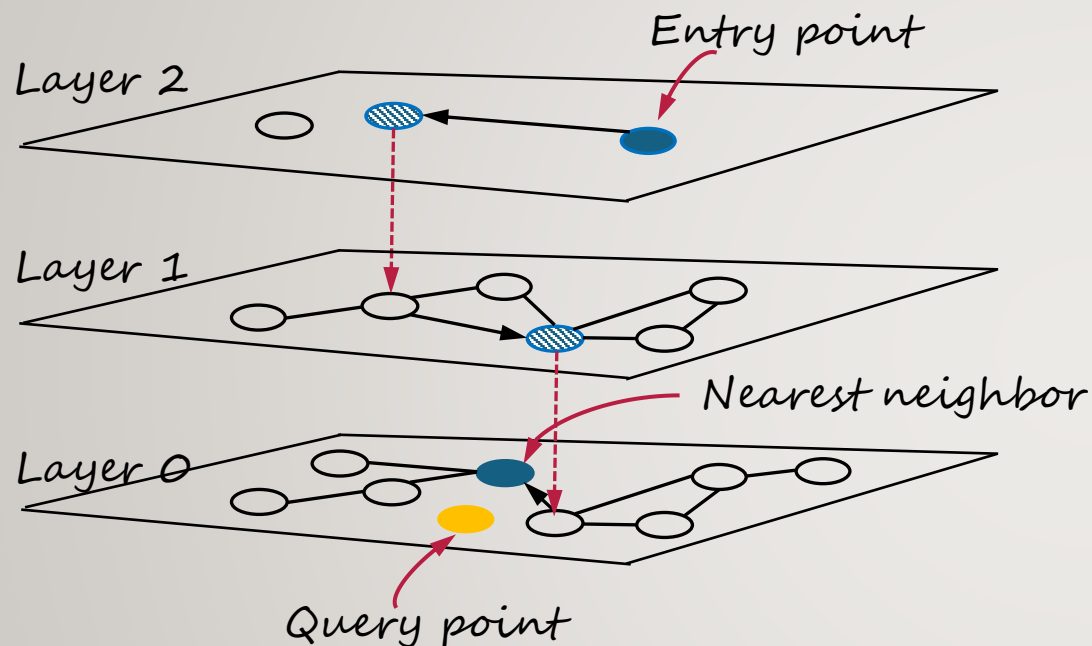


REFINEMENT-BASED CONSTRUCTION (NNDESCENT)



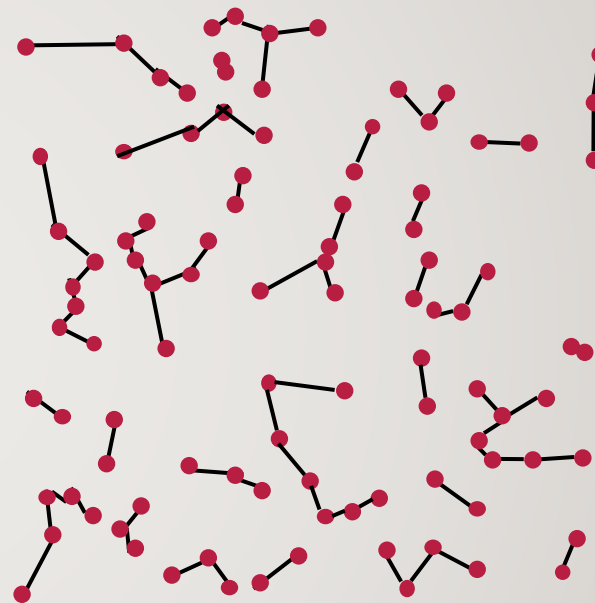
GRAPH-BASED INDEX CONSTRUCTION

INCREMENT-BASED



- **Benefits:** $\sim O(\log N)$ online search, incremental inserts
- **Drawbacks:** Slow construction due to ANNS for each point inserted

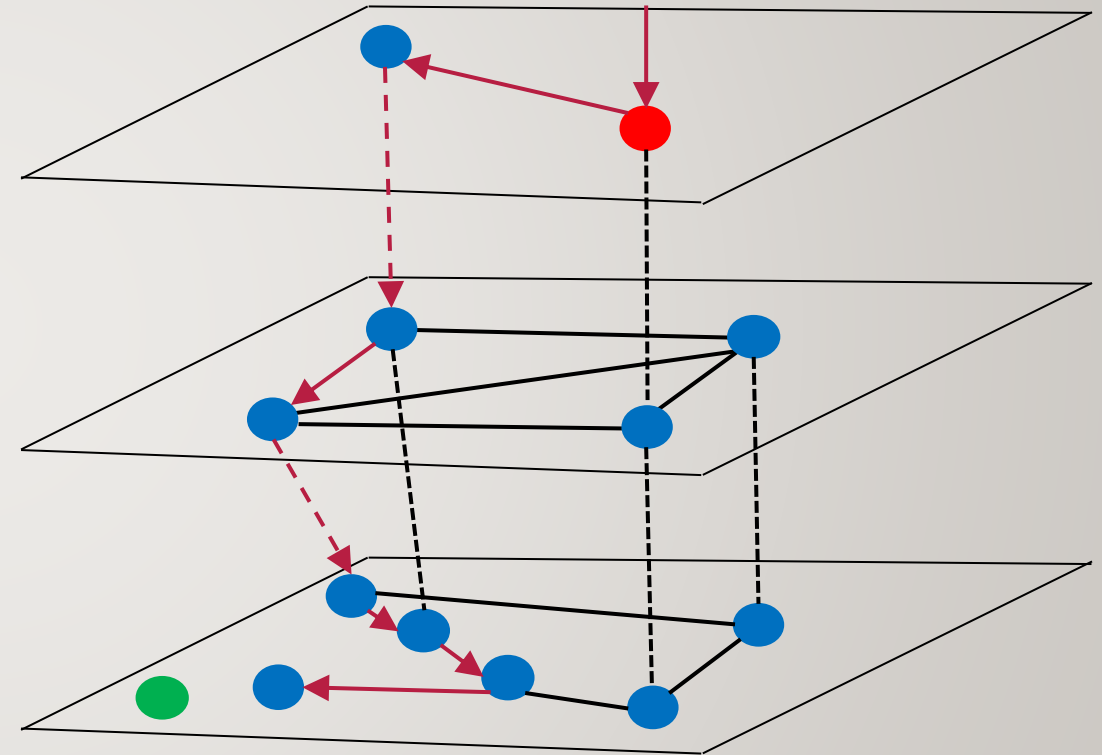
REFINEMENT-BASED



- **Benefits:** Fast construction
- **Drawbacks:** Slow for online queries, no incremental inserts

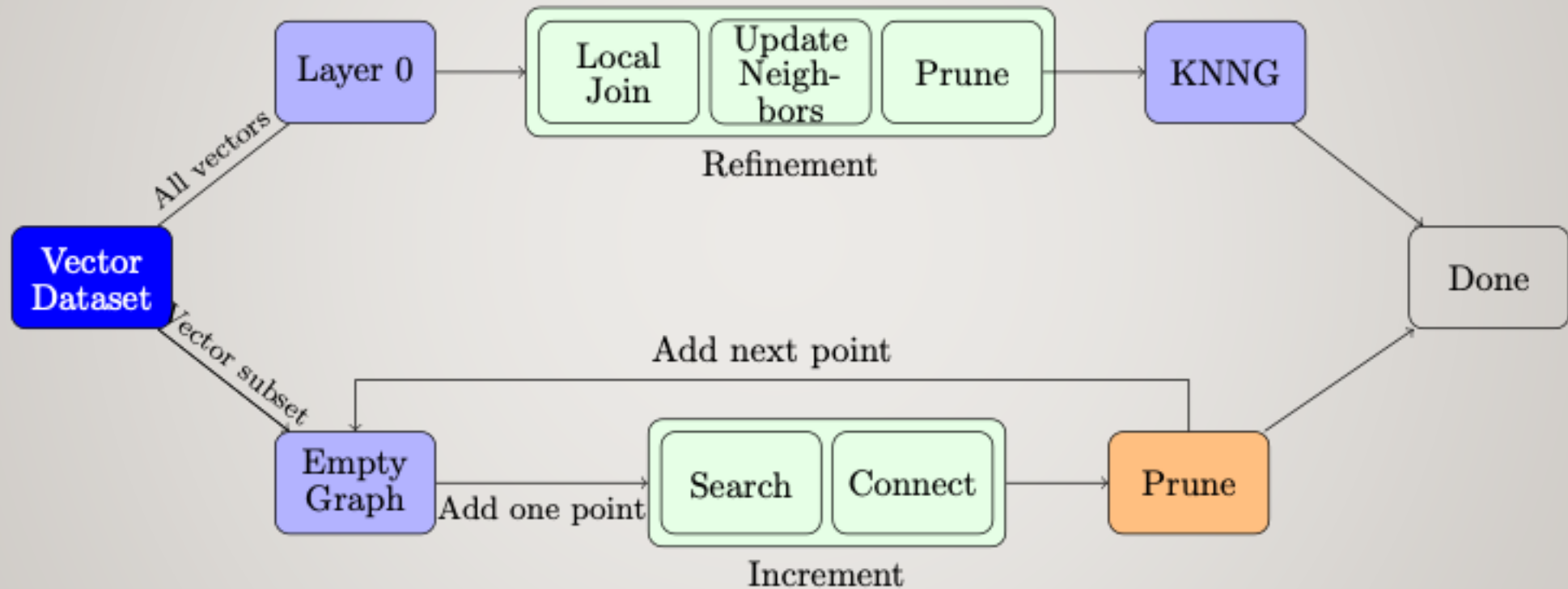
MIRAGE¹

- Combine increment-based and refinement-based construction for both fast construction and search performance
- Multi-layer structure
 - Layer 0 constructed using a refinement-based approach
 - Upper layers constructed using ANNS
 - Layer 0 contains all the vectors
- Implemented in FAISS using C++

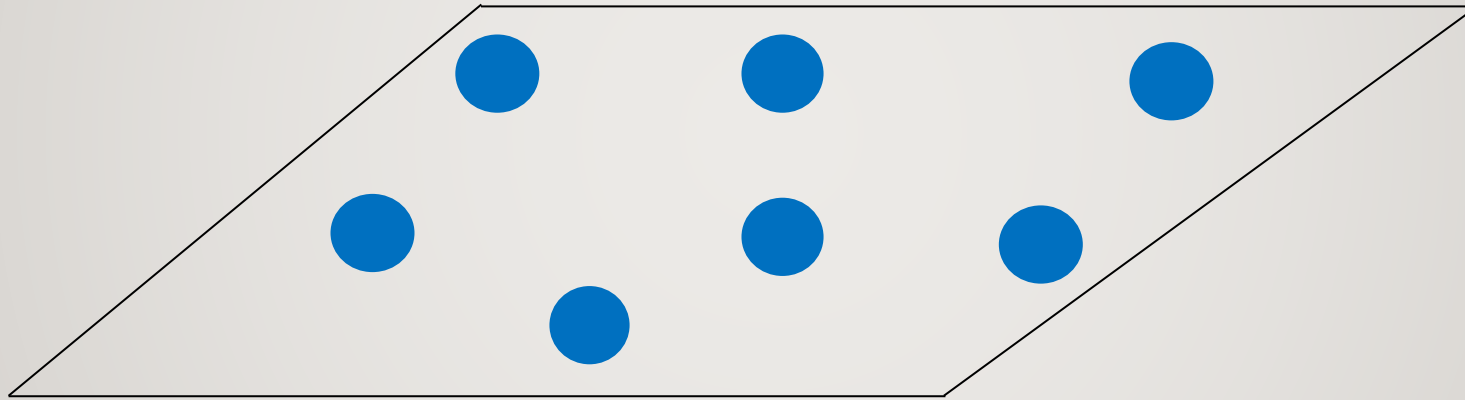


¹Voruganti and Özsu. MIRAGE-ANNS: mixed approach graph-based indexing for approximate nearest neighbor search. *Proc. ACM Management of Data* 3, 3 (2025).

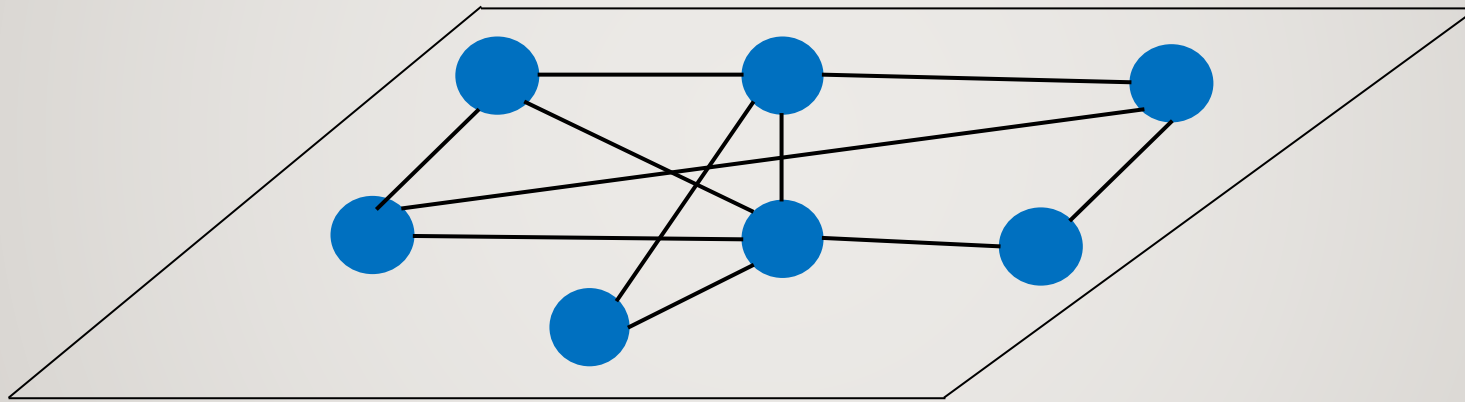
MIRAGE CONSTRUCTION



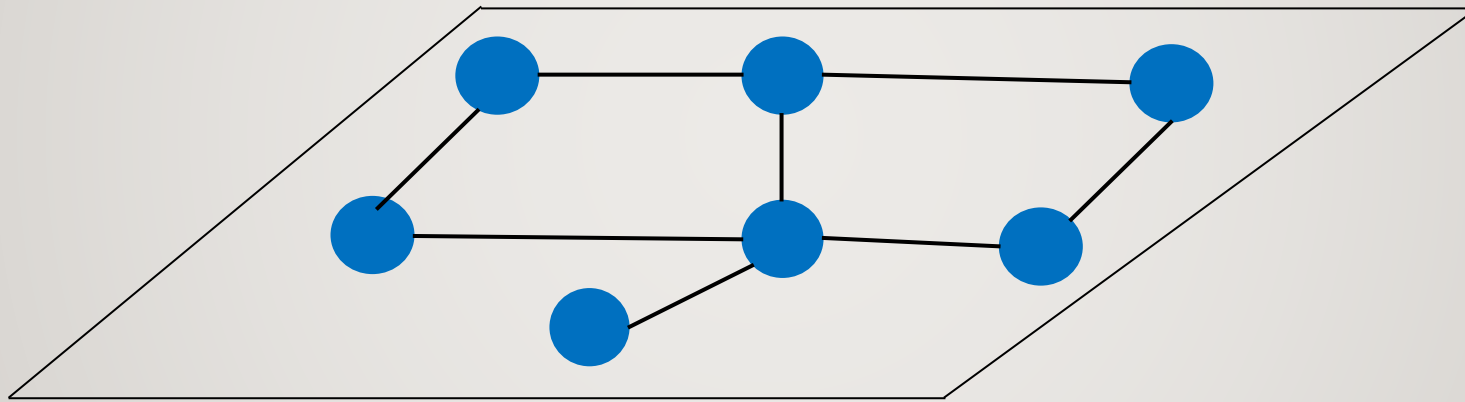
LAYER 0 CONSTRUCTION



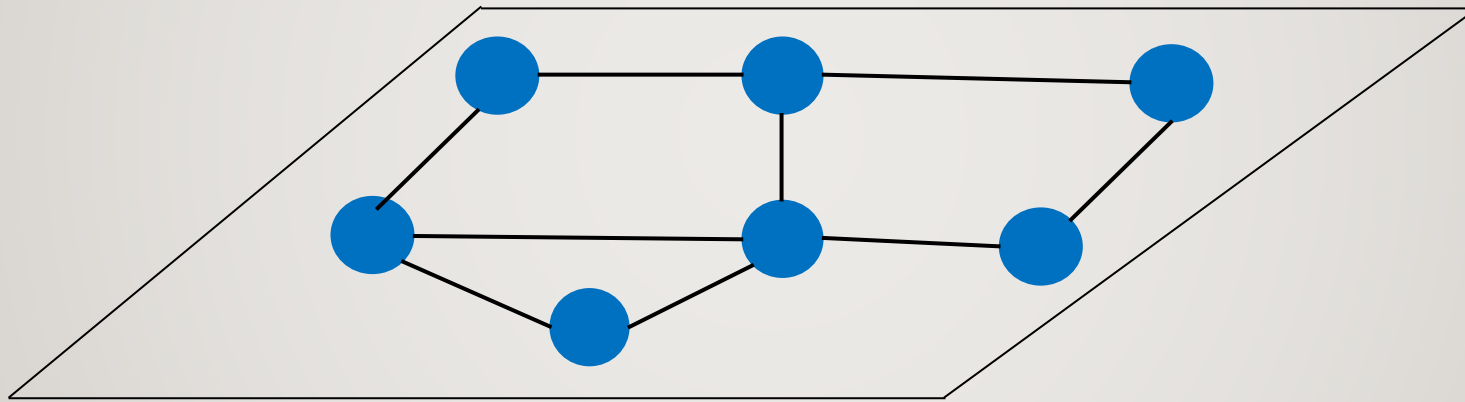
LAYER 0 CONSTRUCTION



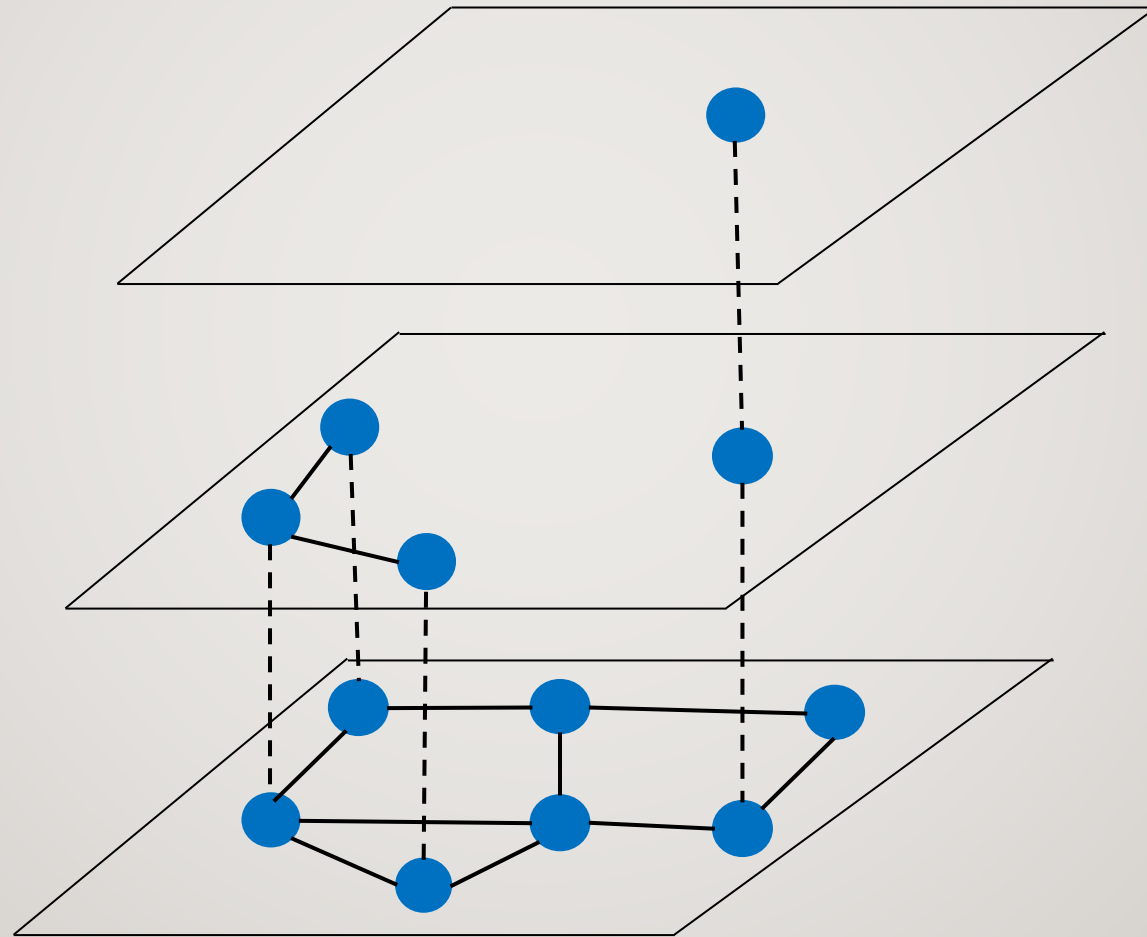
LAYER 0 CONSTRUCTION



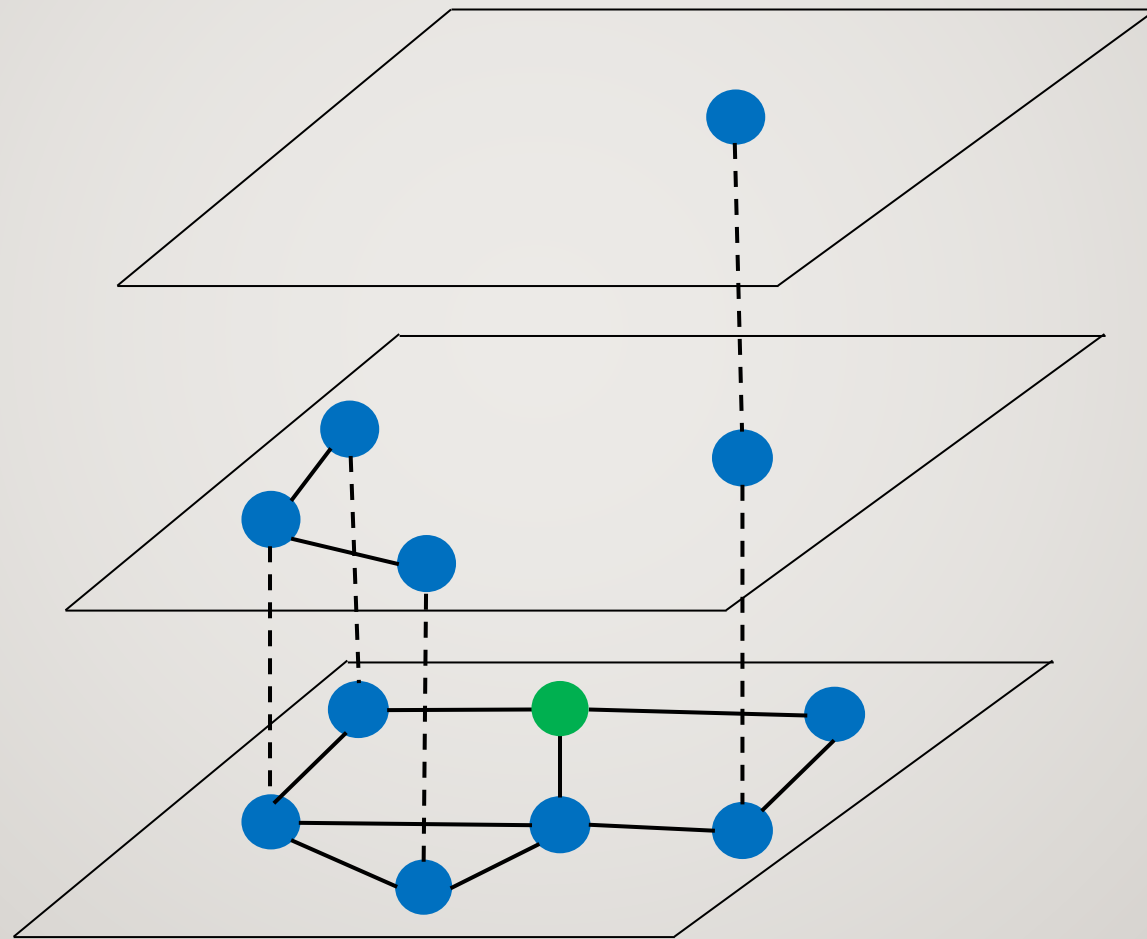
LAYER 0 CONSTRUCTION



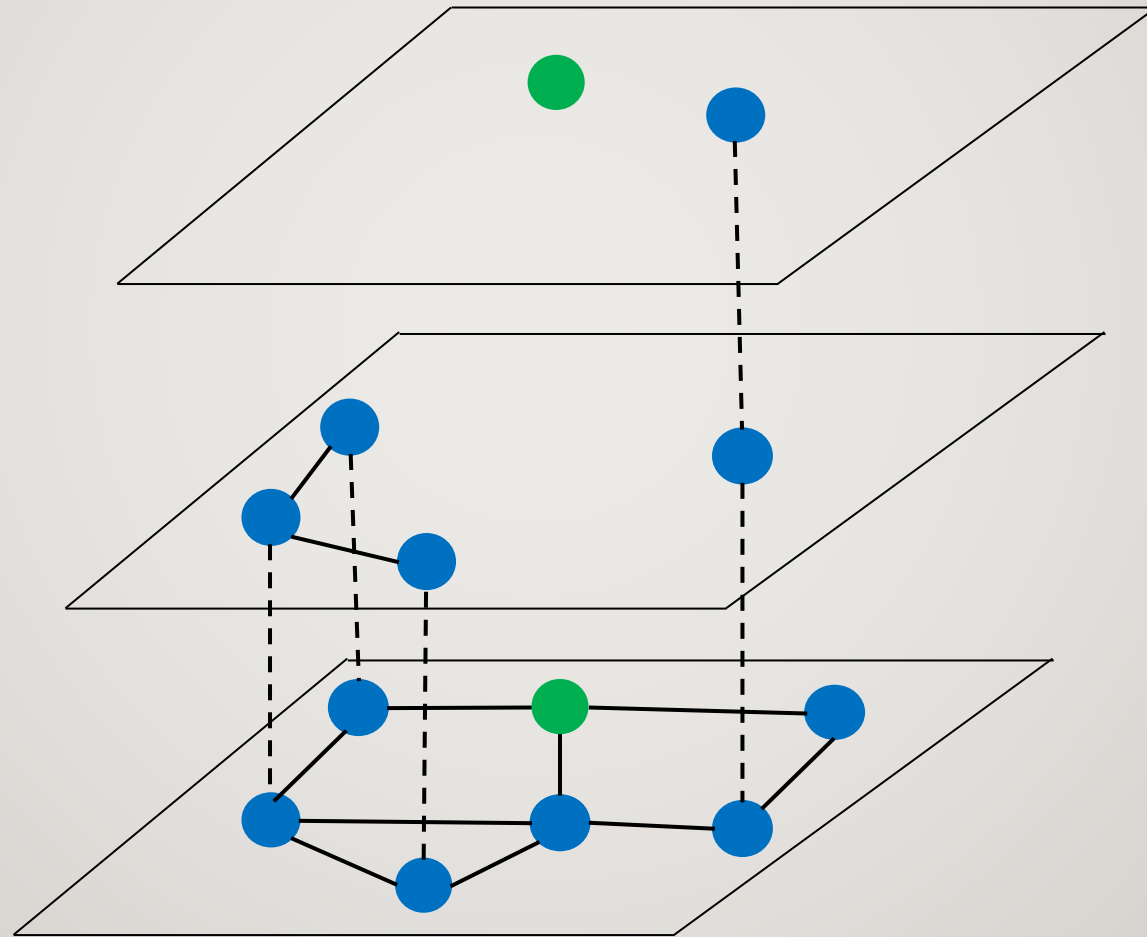
CONSTRUCTION OF UPPER LAYERS



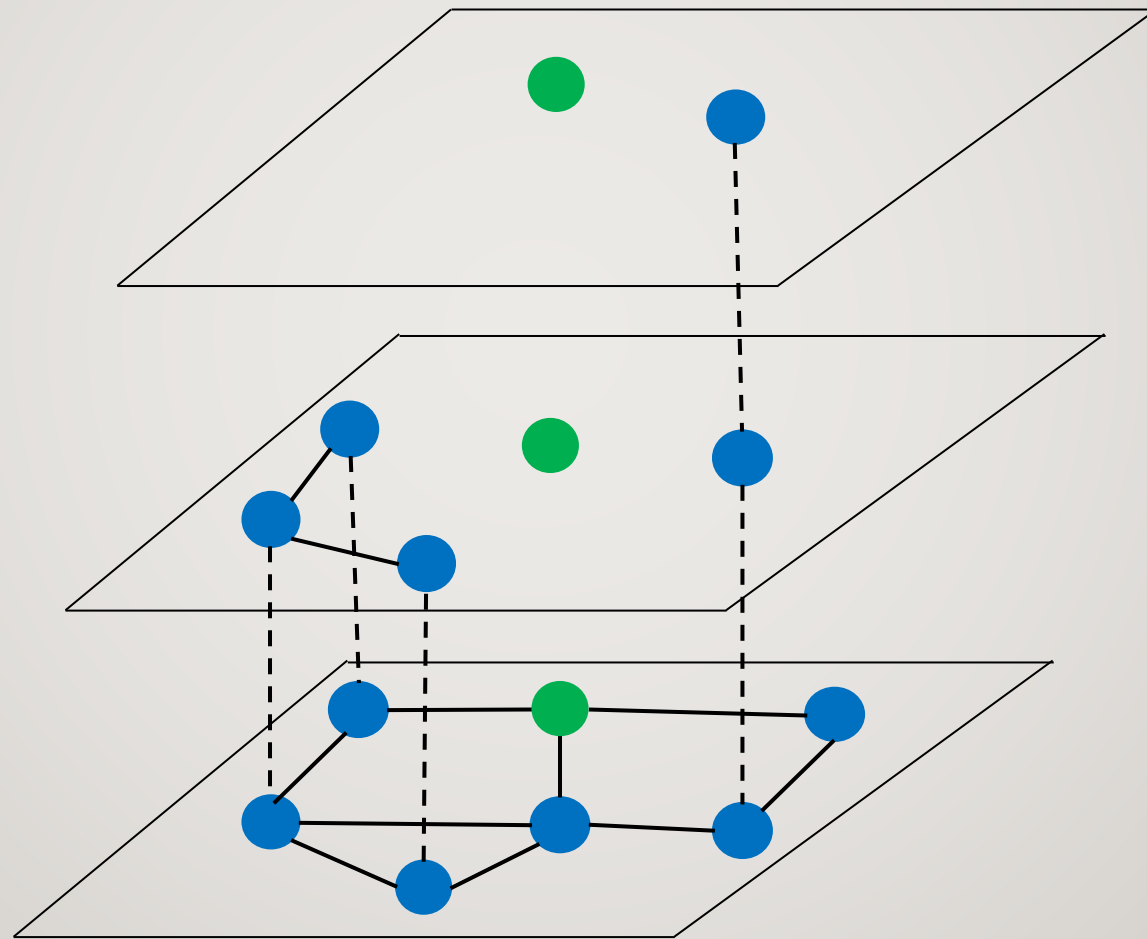
CONSTRUCTION OF UPPER LAYERS



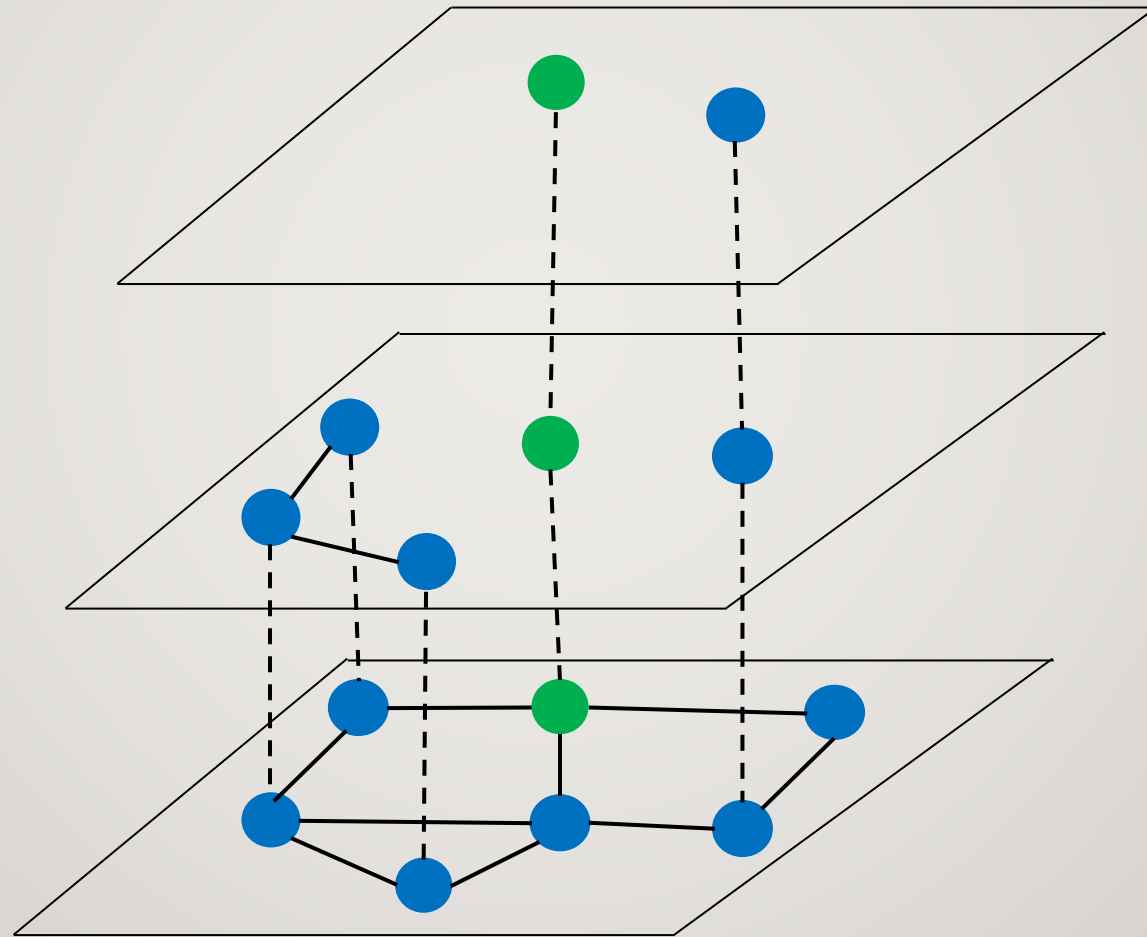
CONSTRUCTION OF UPPER LAYERS



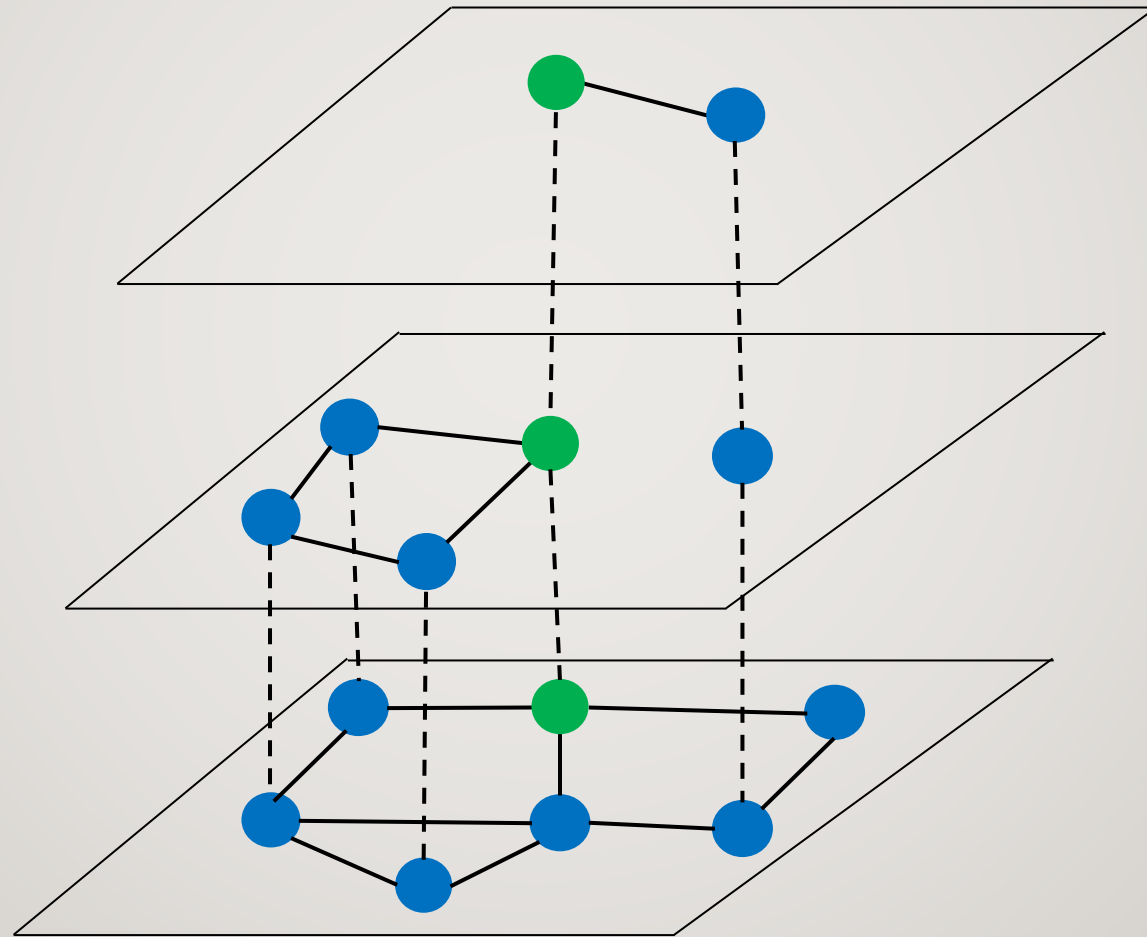
CONSTRUCTION OF UPPER LAYERS



CONSTRUCTION OF UPPER LAYERS



CONSTRUCTION OF UPPER LAYERS



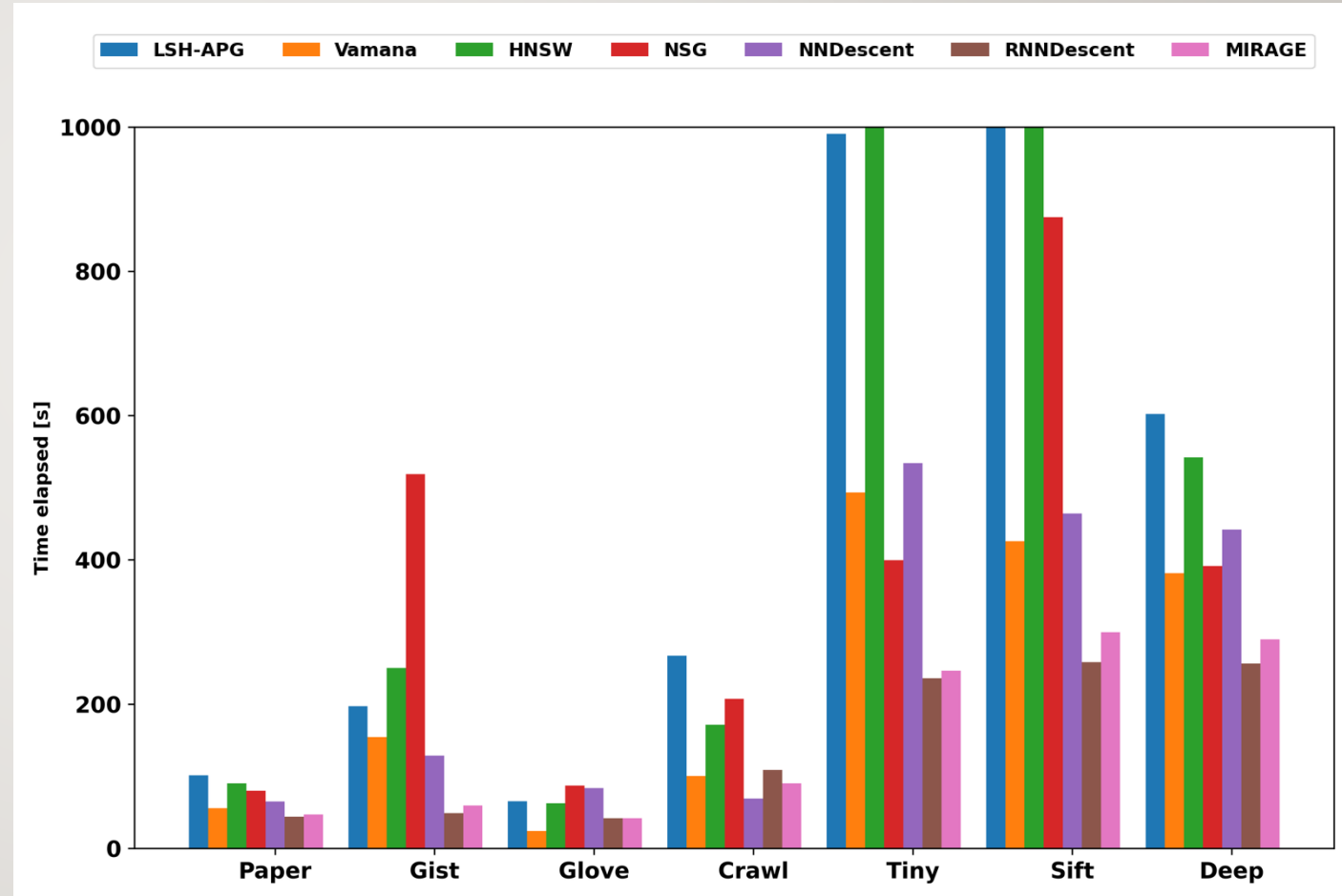
EXPERIMENTAL SETUP

- Test on 7 datasets with different LID values
- LID:
 - Quantifies the effective number of dimensions in a point's local neighborhood.
 - Higher LID means a harder dataset.
- Graph-based indexes tested:
 - HNSW
 - Vamana (DiskANN)
 - LSH-APG
 - NN-Descent
 - RNNDescent
 - NSG

Dataset	Dimension	# Base	# Query	LID	Type
Tiny5M	384	5,000,000	1,000	-	Image
Deep10M	96	10,000,000	10,000	12.1	Image
SIFT10M	128	10,000,000	10,000	9.3	Image
GIST1M	960	1,000,000	1,000	18.9	Image
Crawl	300	1,989,995	10,000	15.7	Text
GloVe	100	1,183,514	1,000	20.0	Text
Paper	200	2,029,997	10,000	-	Text

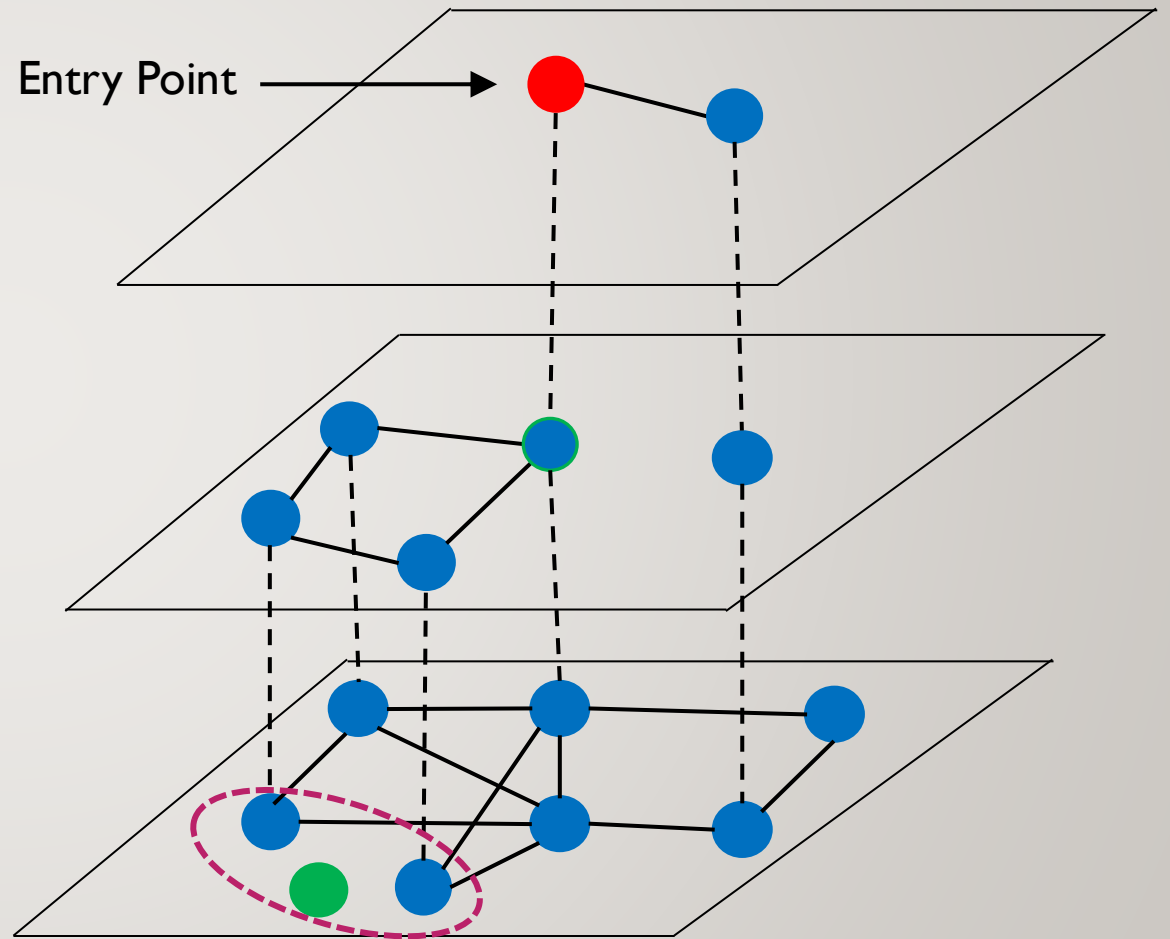
INDEXING TIME

- Up to 6x faster than HNSW and 7x faster than NSG



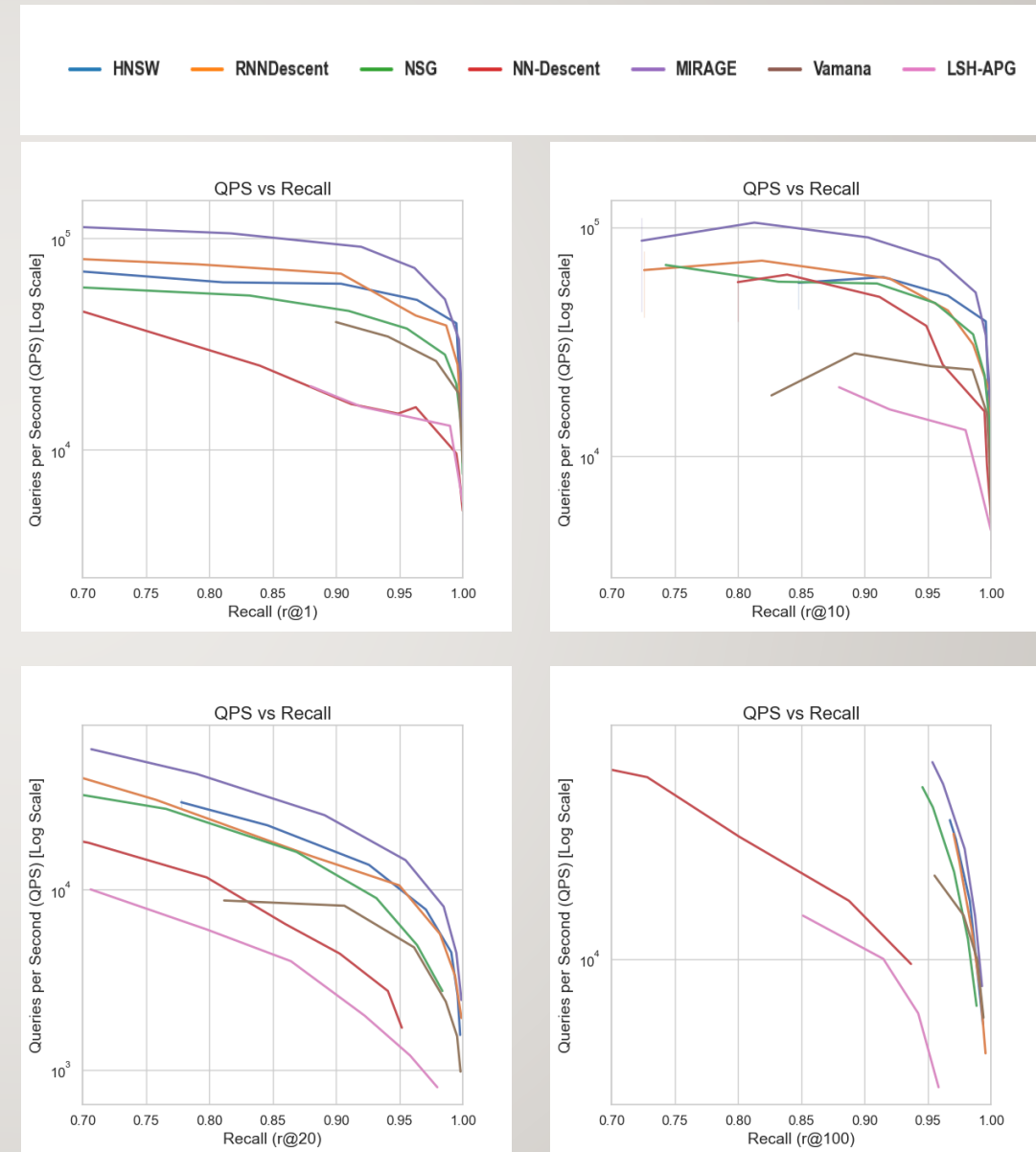
SEARCH STRATEGY

- Queries start at pre-defined point on the highest layer
- Find nearest neighbor to query on each layer
- Conclude after finding K-nearest neighbors at Layer 0
- Hierarchy allows for better entry points to the bottom layer



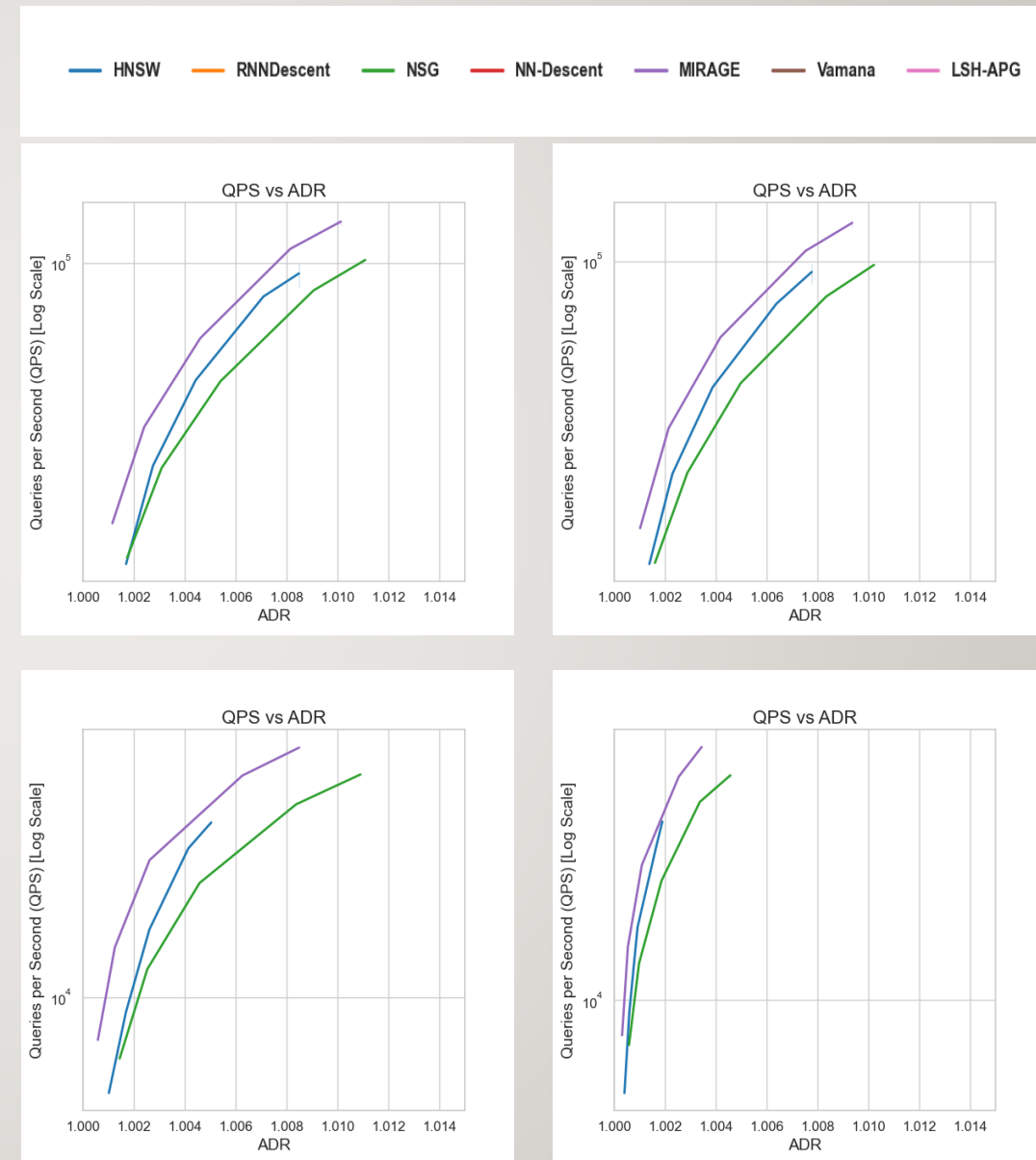
SEARCH PERFORMANCE

- QPS vs Recall
- Test 4 K values
 - $K = 1$
 - $K = 10$
 - $K = 20$
 - $K = 100$
- Optimal search parameters chosen for each index
- Up to 2x QPS as compared to other solutions



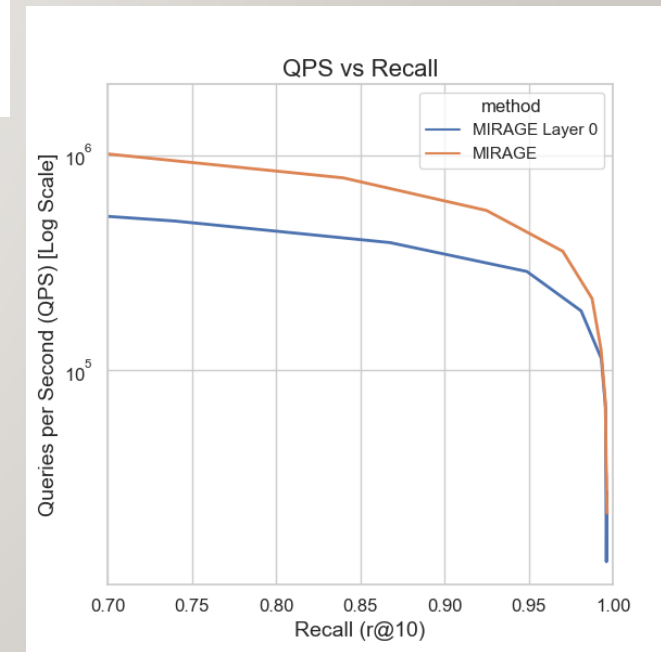
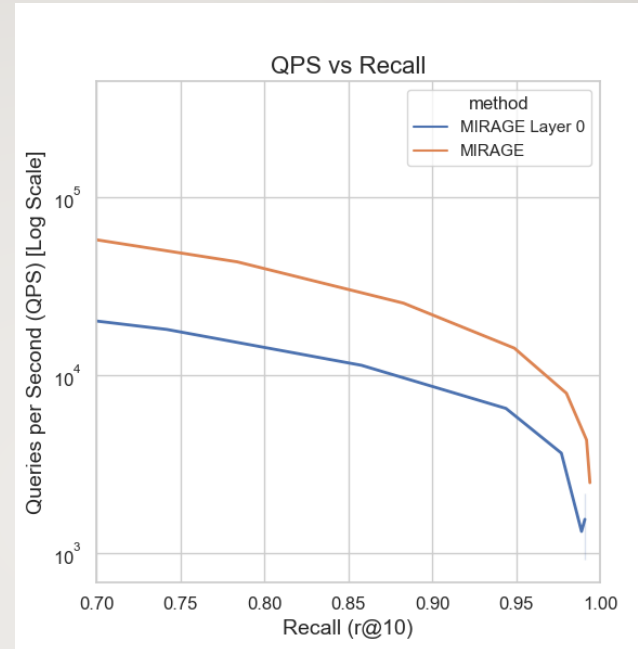
SEARCH PERFORMANCE

- QPS vs ADR
- ADR: How close the retrieved approximate neighbors are to the true nearest neighbors, in terms of distance.
- Test 4 K values on the Crawl and Glove datasets
- Compare MIRAGE against the 3 top performing search solutions



ABLATION STUDY

- Ablation study
 - Fix out-degree and search hierarchical vs Layer 0
- For the non-hierarchical version, we randomly sample entry points
- Results show hierarchy helps QPS



WHY DOES IT WORK?

Graph	Gist		Glove		Crawl		Paper		Sift		Tiny		Deep	
	AOD	IS	AOD	IS	AOD	IS	AOD	IS	AOD	IS	AOD	IS	AOD	IS
HNSW Layer 0	21.81839	3.658	15.892	0.511	13.786	2.326	26.834	1.715	43.678	6.396	26.946	7.654	46.512	5.307
MIRAGE Layer 0	17.424	3.641	15.622	0.510	10.084	2.299	11.036	1.596	24.917	5.697	21.272	7.549	25.145	4.512
NSG	12.531	3.623	9.599	0.483	10.289	2.300	18.612	1.653	27.972	5.810	18.171	7.491	32.825	4.798
RNNDescent	22.127	3.659	20.743	0.532	11.698	2.311	12.955	1.610	26.838	5.768	32.809	7.764	27.536	4.601
NN-Descent	100	3.949	100	0.882	80	2.817	100	2.269	90	8.121	100	9.015	90	6.926
Vamana	67.225	3.83	56.427	0.706	39.754	2.579	66.194	2.061	69.423	7.531	69.283	8.645	69.305	6.35
LSH-APG	32.443	3.785	29.630	0.585	31.059	2.513	35.262	1.821	36.610	6.279	32.990	7.953	35.666	5.022

Table 3: Average vertex out-degree (AOD) and Index size (IS) for each method. Index size is represented in GB.

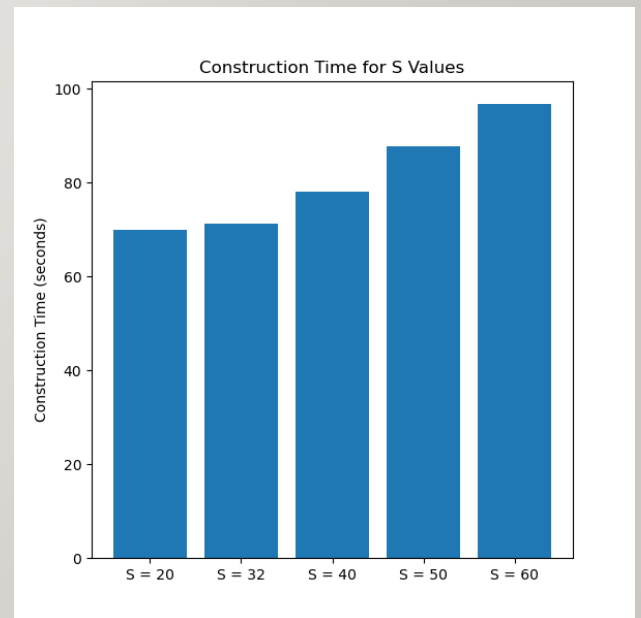
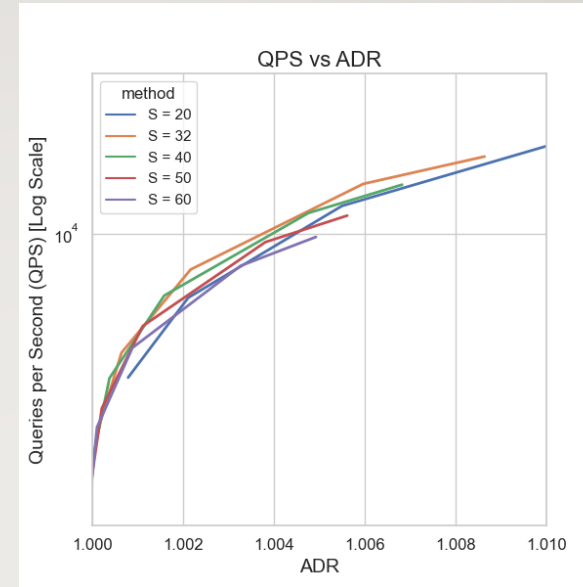
- Sparse lower layer as opposed to HNSW
- Reduces distance computations on the lower layer

Table 4: Number of distance computations to achieve 0.99 recall with $k = 100$. Number in parenthesis represents the difference between HNSW and MIRAGE.

	HNSW	MIRAGE
Paper	3936 (2.4× greater)	1588
Crawl	4319 (2.4× greater)	1758
Gist	25161 (1.7× greater)	14776
Sift	1474 (2× greater)	716

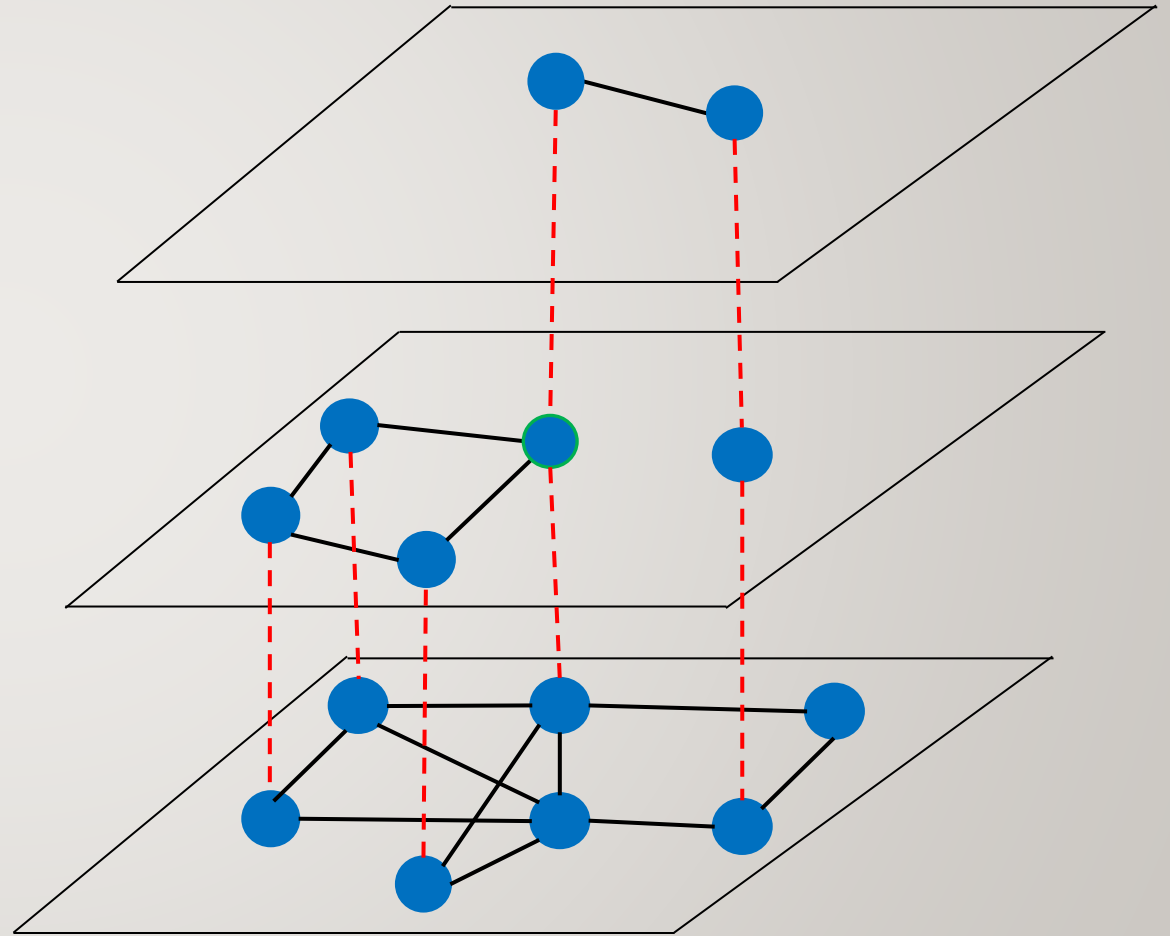
TRADEOFF- ACCURACY AND QPS

- The denser the lower layer, the less accuracy degradation
- Tradeoff: Less QPS
- Denser lower layer also increases construction time



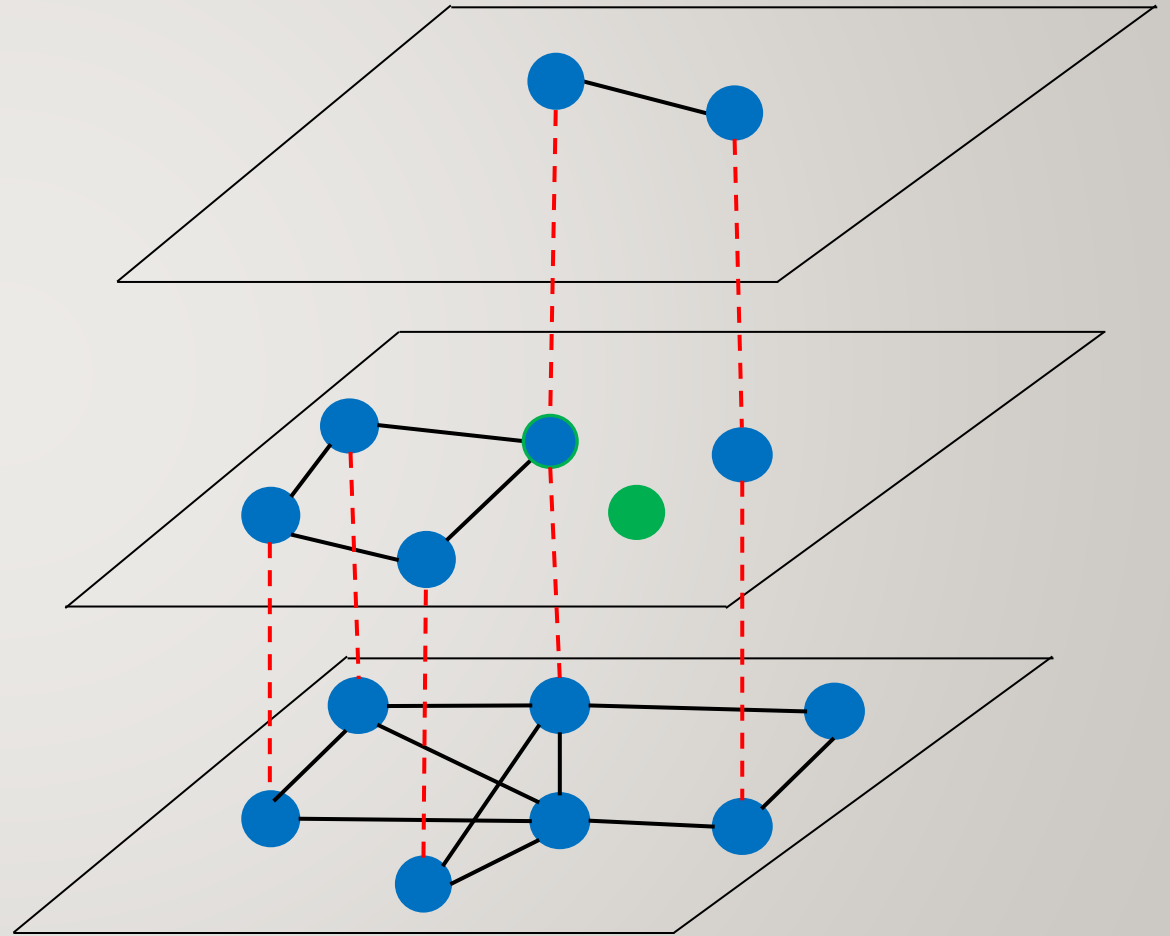
INCREMENTAL INSERTS

- New inserts use an increment-based approach, treating each point as a query and performing ANNS to find neighbors
- Helps maintain graph quality



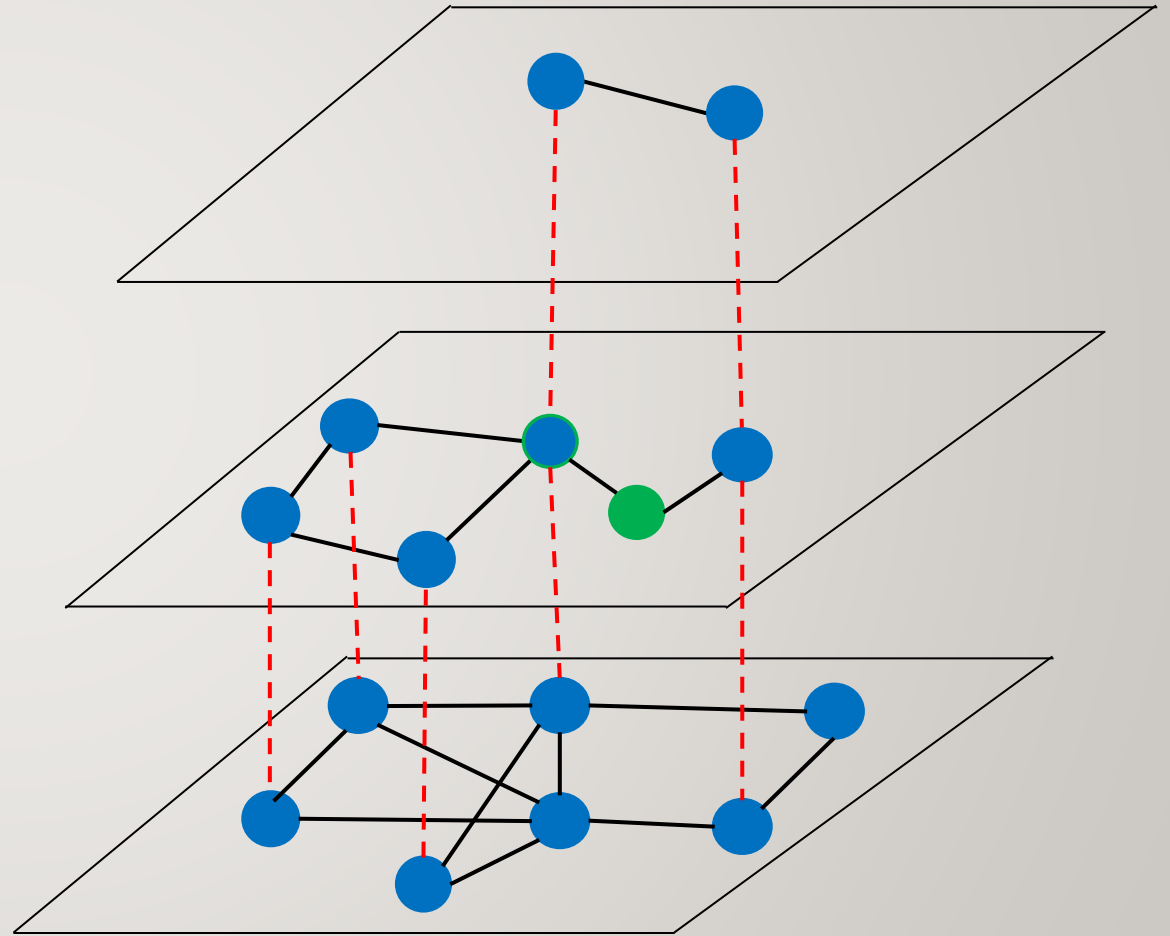
INCREMENTAL INSERTS

- New inserts use an increment-based approach, treating each point as a query and performing ANNS to find neighbors
- Helps maintain graph quality



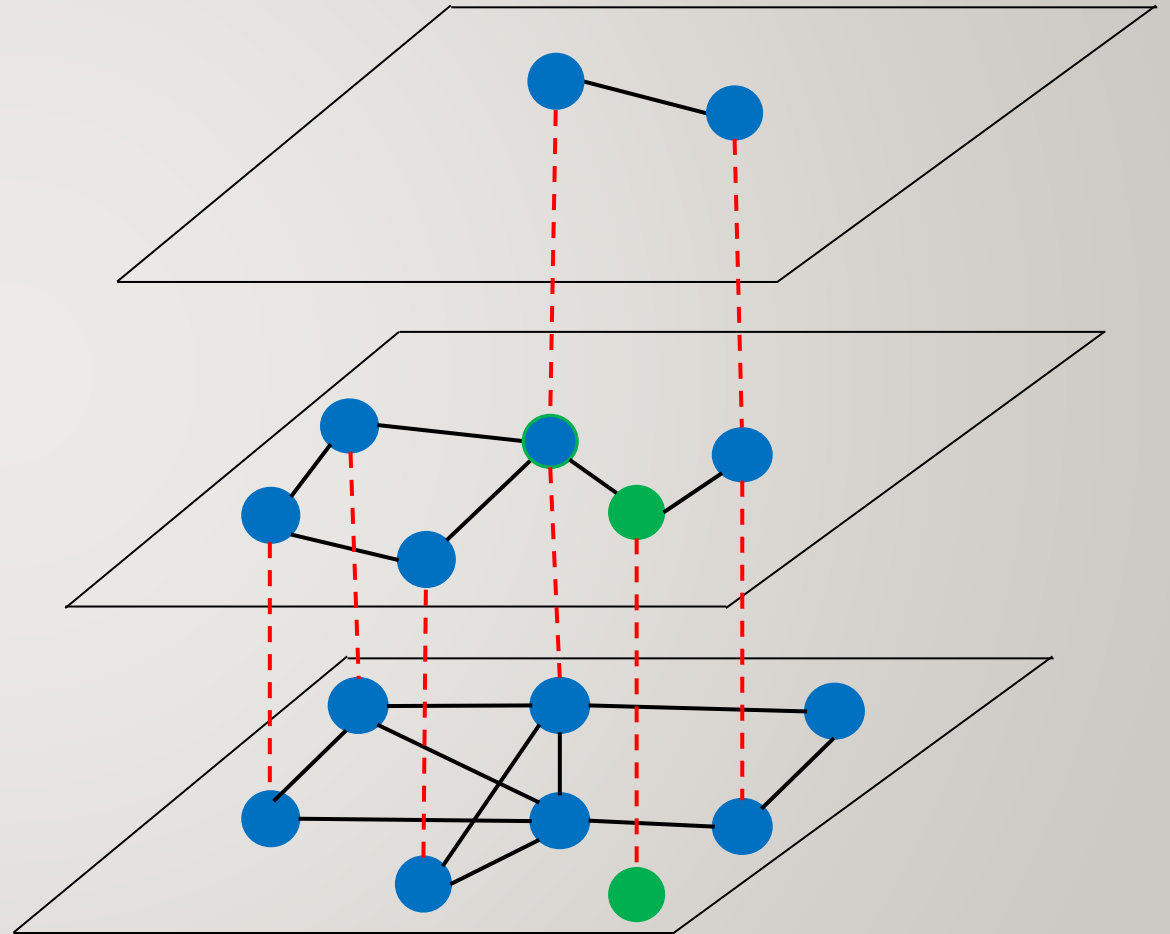
INCREMENTAL INSERTS

- New inserts use an increment-based approach, treating each point as a query and performing ANNS to find neighbors
- Helps maintain graph quality



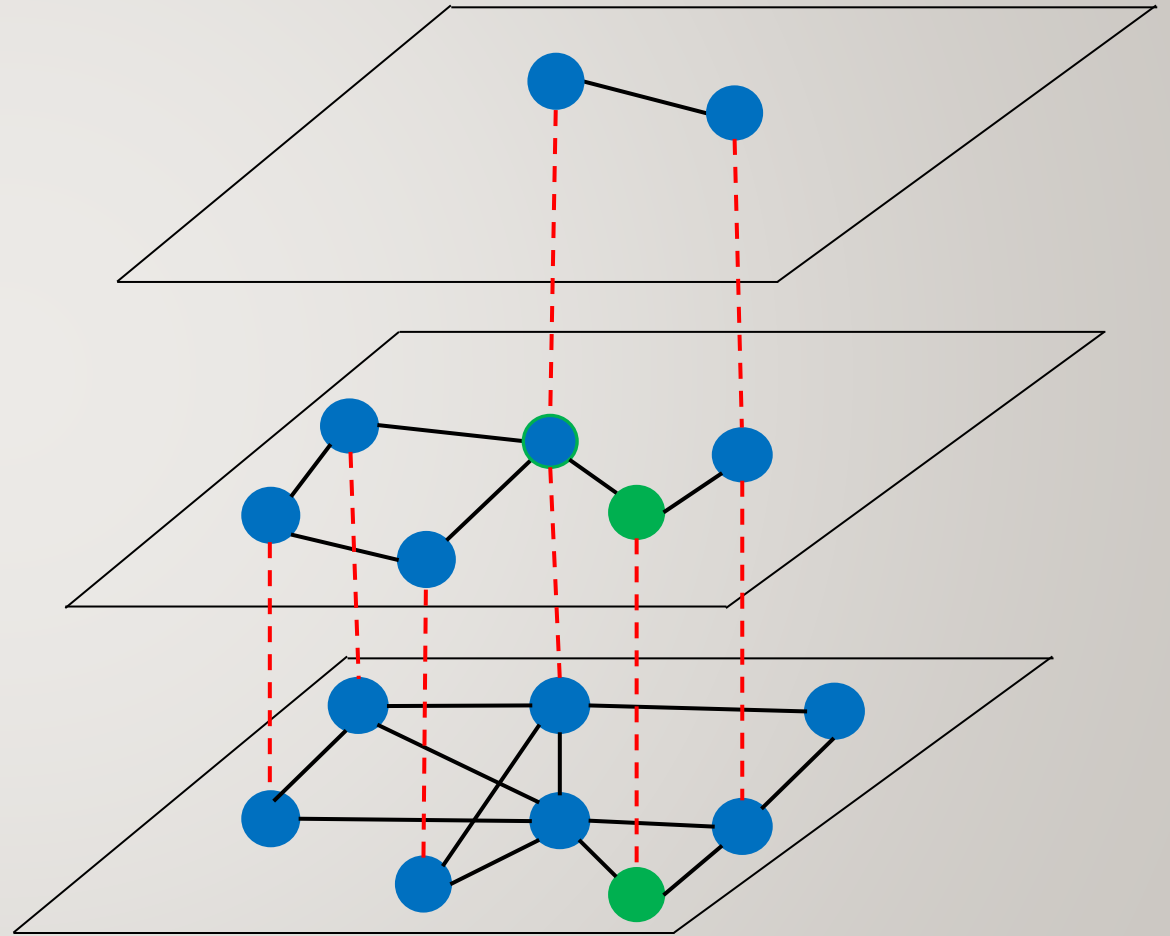
INCREMENTAL INSERTS

- New inserts use an increment-based approach, treating each point as a query and performing ANNS to find neighbors
- Helps maintain graph quality



INCREMENTAL INSERTS

- New inserts use an increment-based approach, treating each point as a query and performing ANNS to find neighbors
- Helps maintain graph quality



TAKE-HOME MESSAGES

- Vectors generated by embedding models are important and becoming prevalent.
- You may not want a vector DBMS, but you will need to search over these.
 - Indexing is critical to performance
- There are ways to achieve high performance with good search quality

Thank
you

