Timestamp Approximate Nearest Neighbor Search over High-Dimensional Vector Data

Yuxiang Wang [1], Ziyuan He [1], Yongxin Tong [1], Zimu Zhou [2], Yiman Zhong [1]

[1] Beihang University

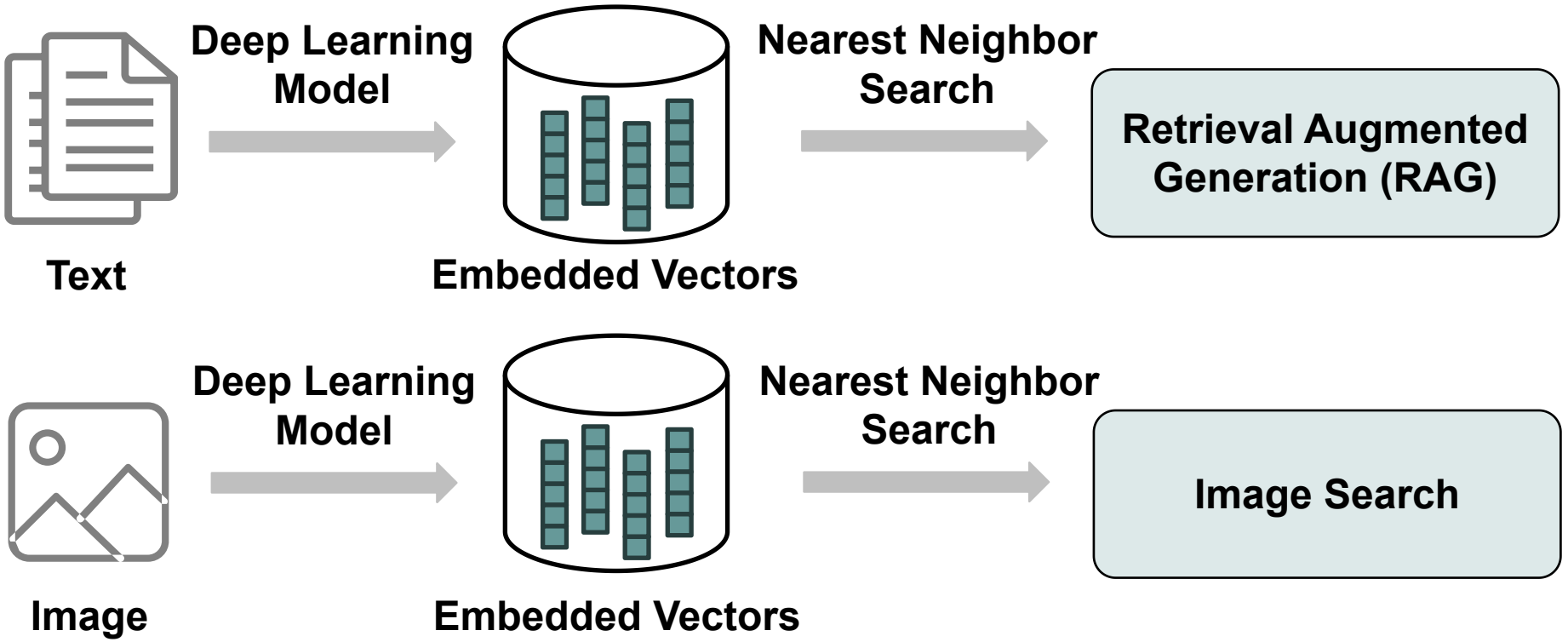
[2] City University of Hong Kong

Outline

- Background and Motivation
- Problem Statement
- Method
- Experiments
- Conclusions

Background

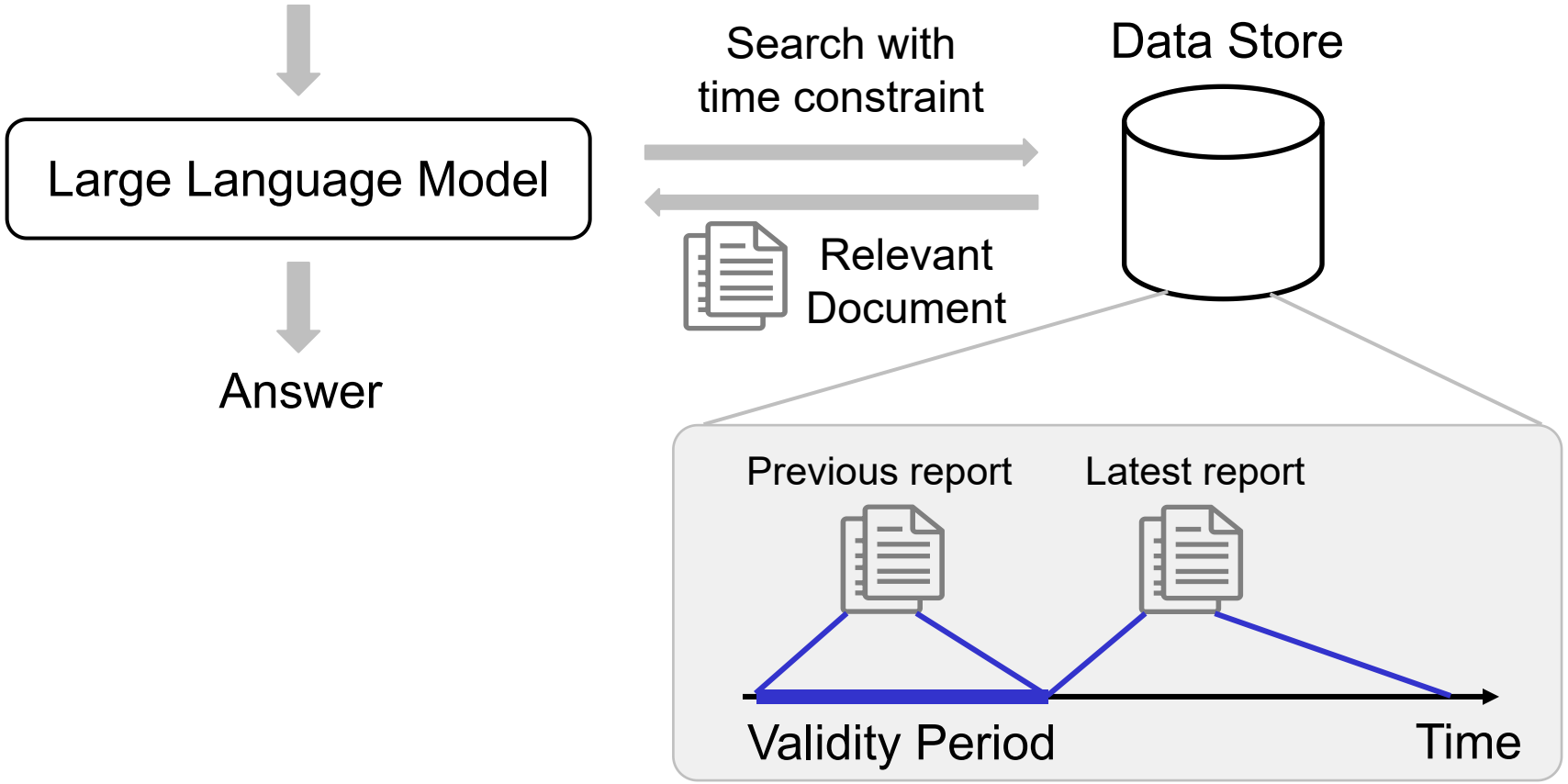
- Unstructured data are increasingly represented as **high-dimensional vectors** for emerging AI applications



Motivation

- Example: RAG for Time-relevant QA

Question: What was the highest closing value of the NASDAQ-100 index before **October 2024**?

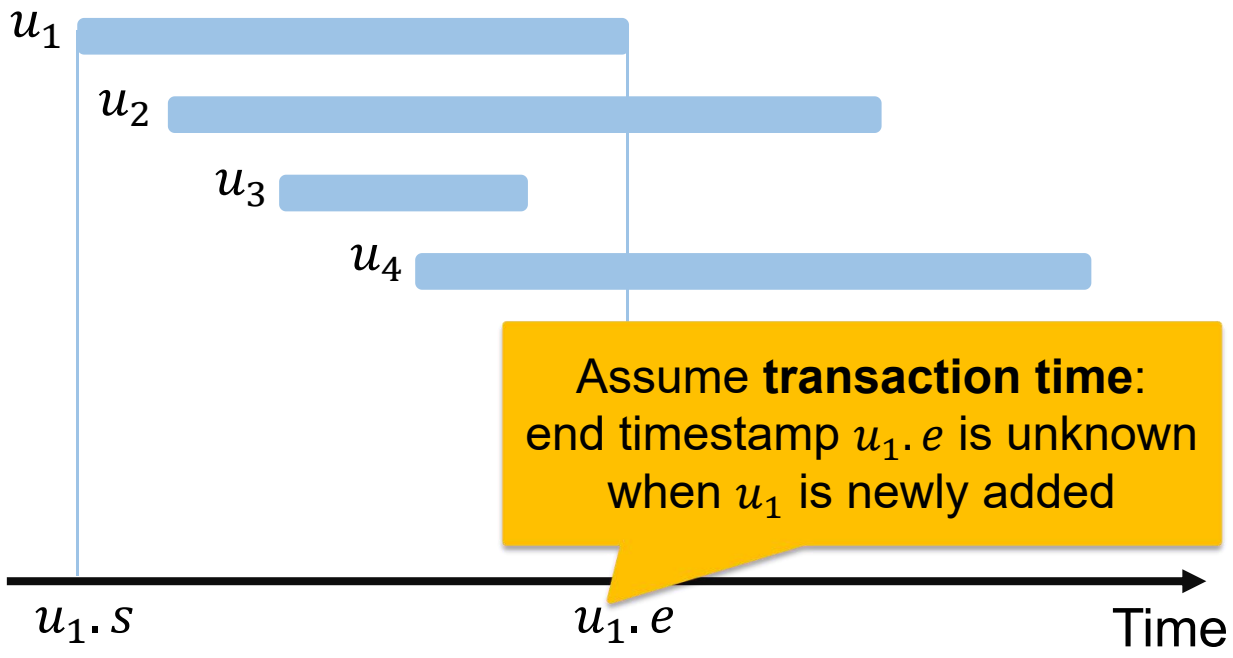


Outline

- Background and Motivation
- Problem Statement
- Method
- Experiment
- Conclusion

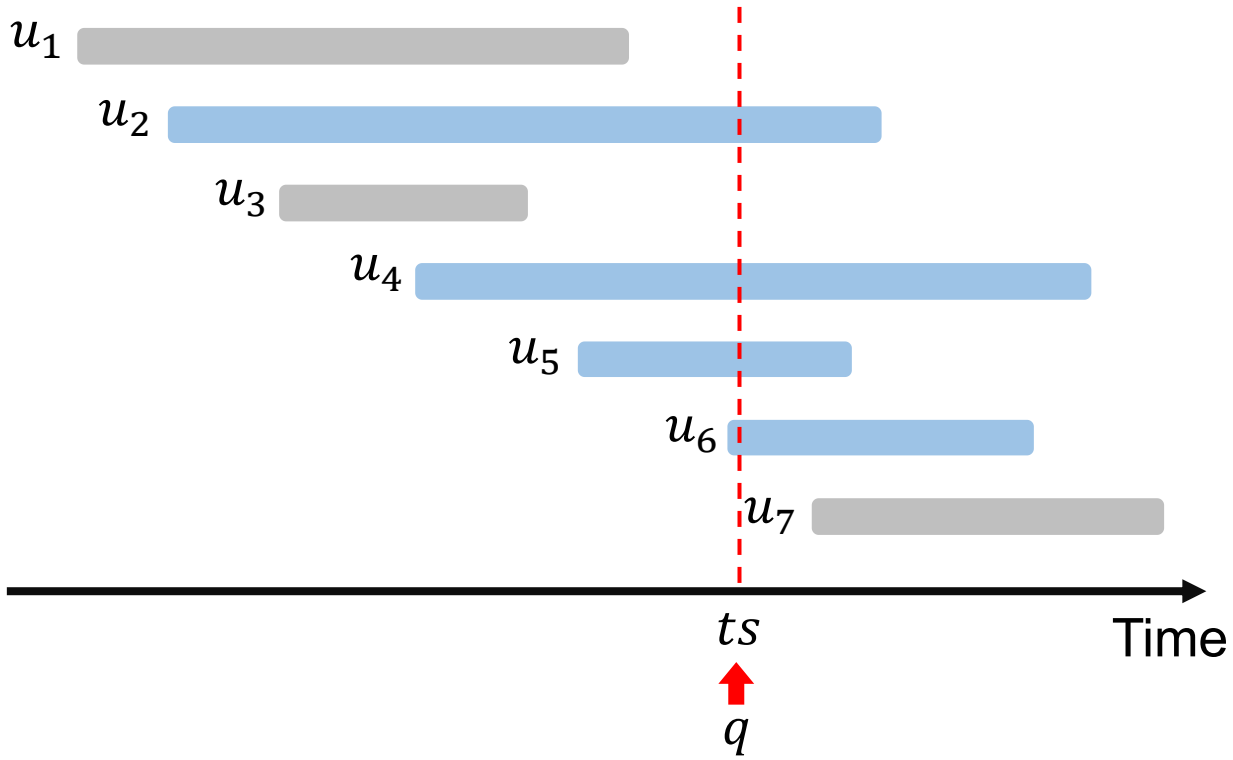
Problem Statement

- Vector Dataset $D = \{u_1, u_2, ...u_N\}$ $(u_i \in \mathbb{R}^d)$
  - Each vector is associated with two timestamps, $u_i.s$ and $u_i.e$, representing the start and end of its valid period



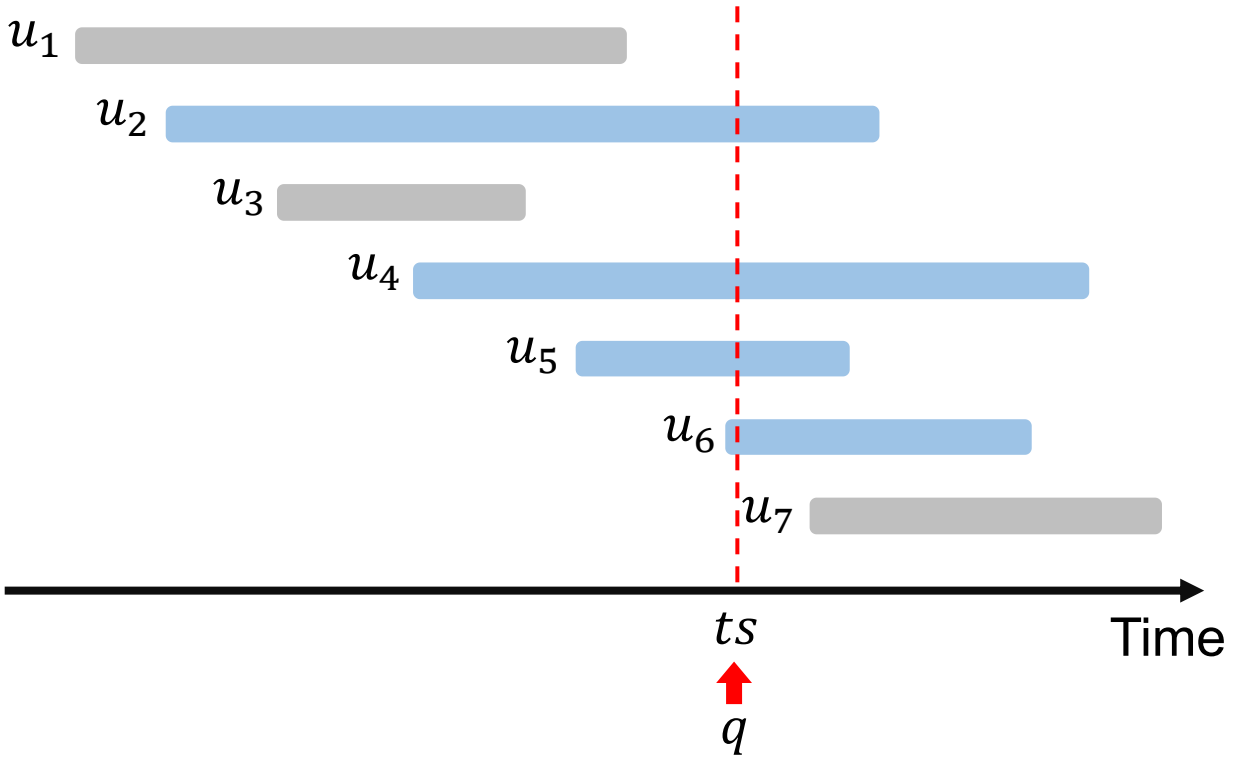
Problem Statement

- Timestamp Approximate Nearest Neighbor Search $TANNS(D, q, ts,\ k)$
  - Find a subset of k valid vectors that are approximately closest to q



Problem Statement

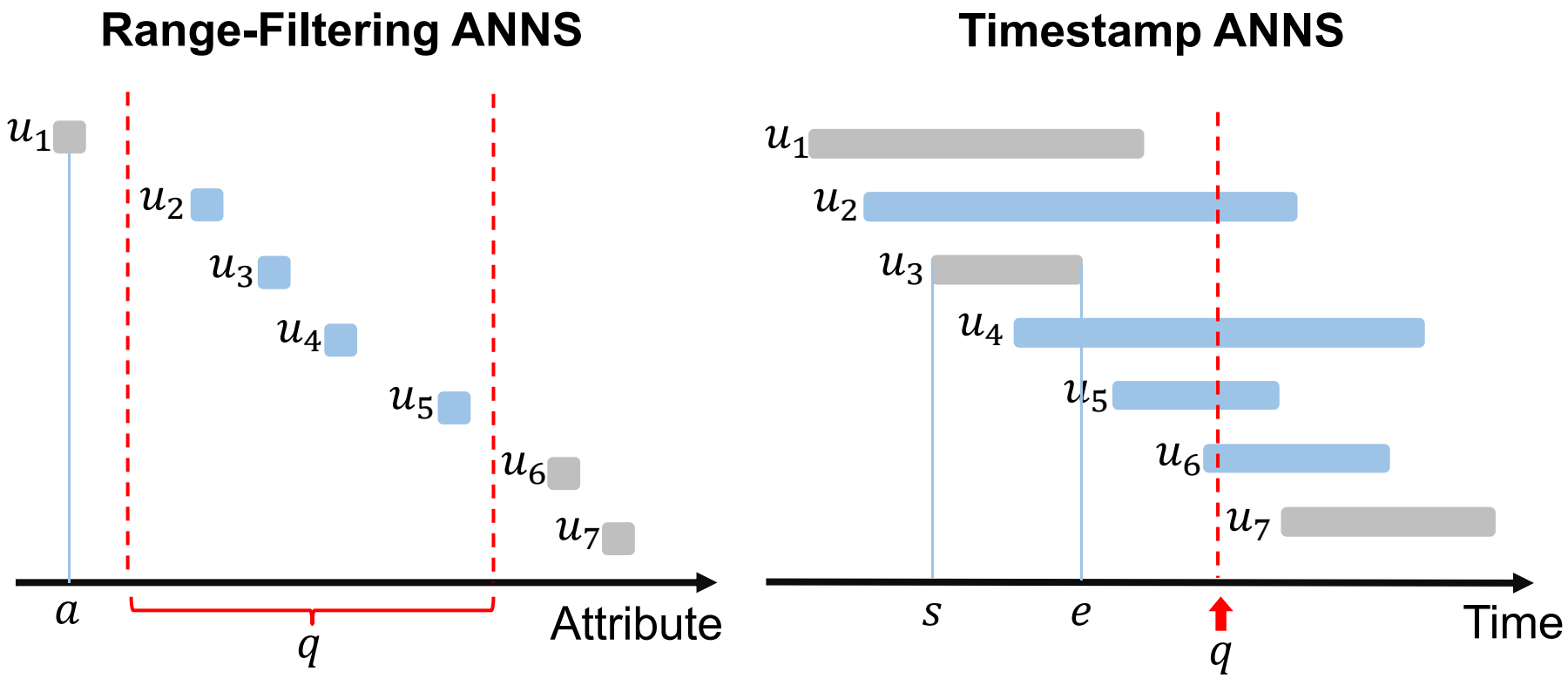
- Example
  - The ground truth for $TANNS(D,\ q, ts, 3)$ is $\{u_4, u_2, u_6\}$



Distance	q
u_1	2.5
u_2	1.9
u_3	2.1
u_4	1.1
u_5	3.2
u_6	2.6
u_7	3.7

Existing Work

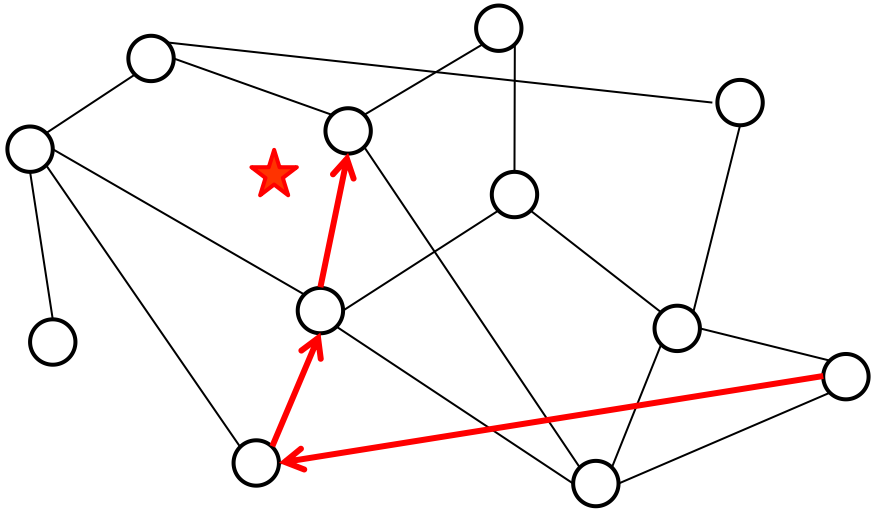
- Differences from Range-Filtering ANNS [1,2,3,4]



[1] SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. SIGMOD 2024.
[2] Approximate Nearest Neighbor Search with Window Filters. ICML 2024.
[3] iRangeGraph: Improvising Range-dedicated Graphs for Range-filtering Nearest Neighbor Search. SIGMOD 2025.
[4] UNIFY: Unified Index for Range Filtered Approximate Nearest Neighbors Search. VLDB 2025.

Preliminaries: Graph Index

- Graph-based indexes are widely adopted for efficient ANNS [1,2]
- Index construction: proximity graph with vectors as points

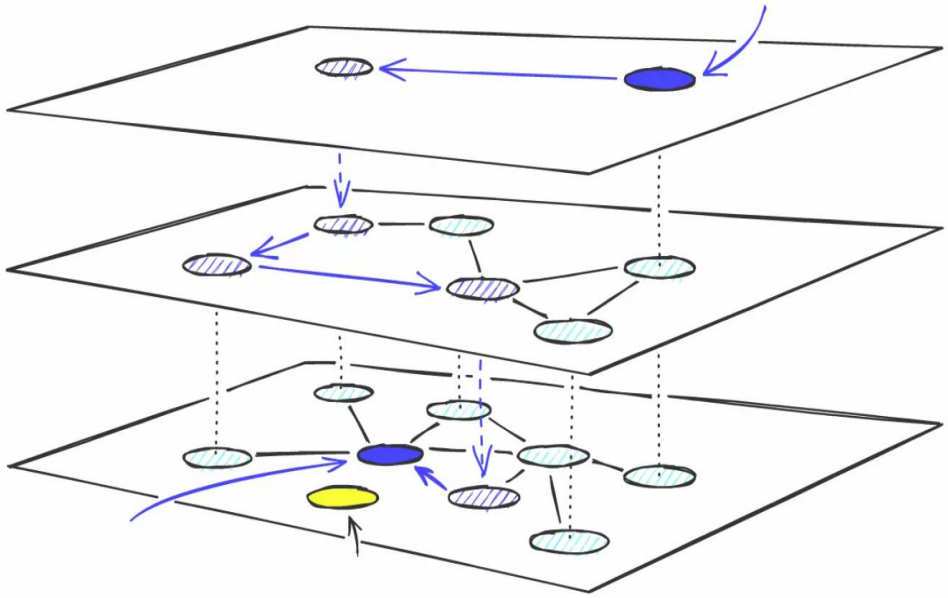


- Search: greedy routing in the graph

[1] A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. PVLDB 2021.
[2] Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. SIGMOD 2025.

Preliminaries: Graph Index

- Hierarchical Navigable Small-World graph (HNSW) [1]
  - Maintains up to M neighbors for each point
  - Achieve high accuracy by visiting $O(\log N)$ points
  - Complexity
    - Search: $O(M\log N)$
    - Space: $O(MN)$



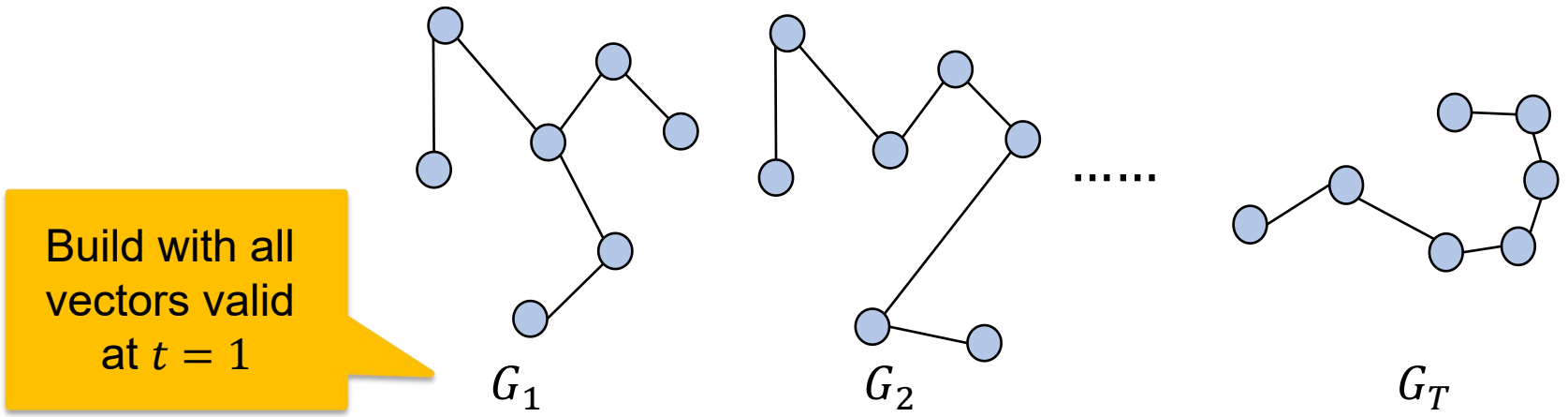
[1] Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. TPAMI 2018.

Outline

- Background and Motivation
- Problem Statement
- Method
- Experiments
- Conclusions

Naïve Graph-based TANNS

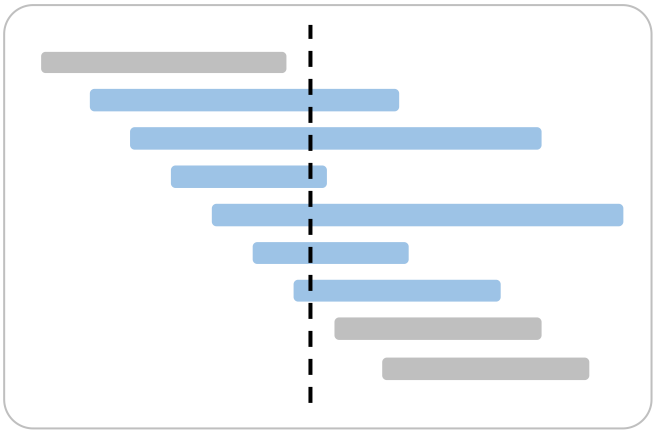
- Build graph index for vectors valid at each timestamp, resulting in T graph



- Given query with timestamp ts, search in G_{ts} with greedy routing
- Incur a large space overhead of $O(MN^2)$

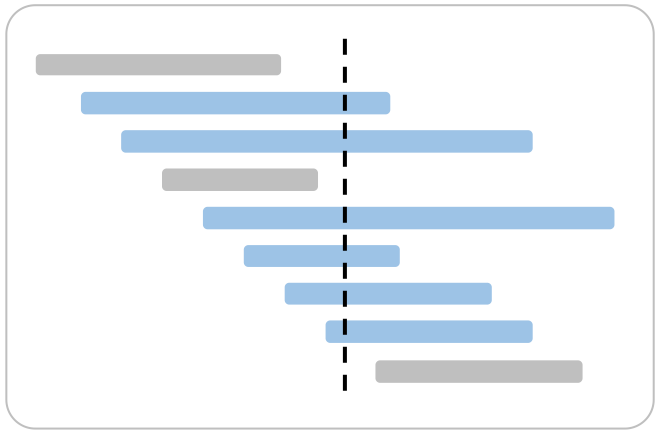
Timestamp Graph

- Temporal Locality: valid vector sets at nearby timestamps largely overlap

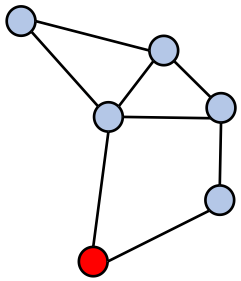


t

Valid data sets are similar

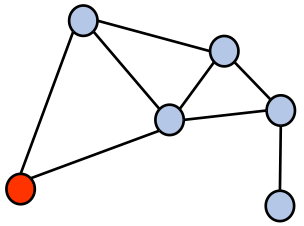


$t + \Delta$



G_t

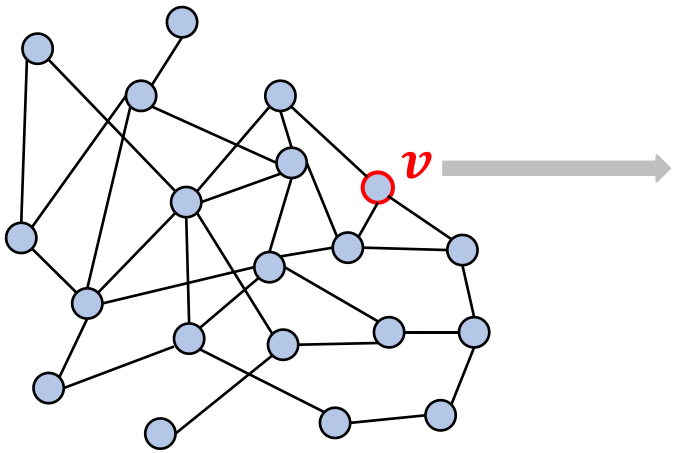
Graph indexes for valid data are similar



$G_{t+\Delta}$

Timestamp Graph

- Construction: aggregate the all per-timestamp graph indexes into a single graph TG
  - For each point, record its neighbor list when the neighbor list change
  - At timestamp t, build G_t on the basis of G_{t-1} and merge the information of G_t into TG



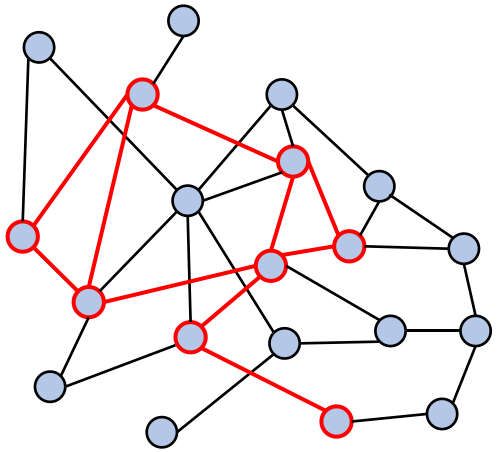
Historic Neighbor List

Timestamp	Neighbors of v
t_1	u_1, u_2, u_3, u_4, u_5
t_2	u_1, u_2, u_3, u_4, u_6
t_3	u_3, u_4, u_6, u_7

At timestamp between t_1 and t_2, neighbors for v are $\{u_1, u_2, u_3, u_4, u_6\}$

Timestamp Graph

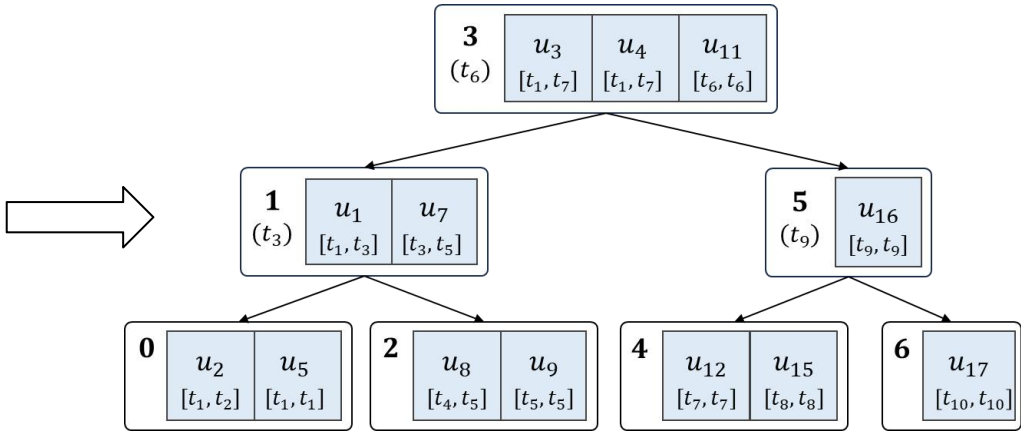
- Search: timestamp-aware greedy routing
  - Decide neighbor list of a point based on the query timestamp
  - When searching with timestamp t, all points visited are valid at t



Compressed Timestamp Graph

- Observation: a point can appear in neighbor lists for multiple timestamps

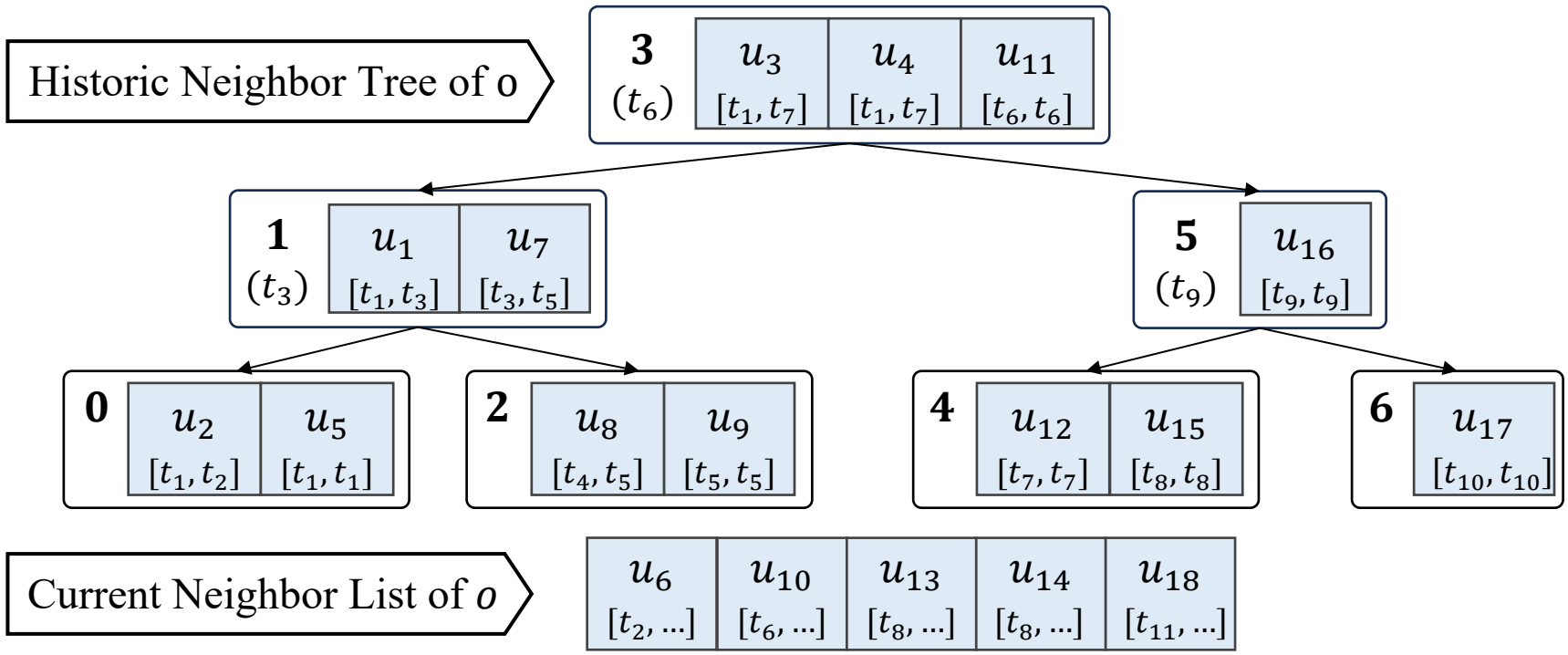
Timestamp	Neighbors of v
t_1	u_1, u_2, u_3, u_4, u_5
t_2	u_1, u_2, u_3, u_4, u_6
t_3	u_1, u_3, u_4, u_6, u_7
t_4	u_3, u_4, u_6, u_7, u_8
……	……



- Solution: compress the neighbor list of each point into tree structure to reduce redundancy

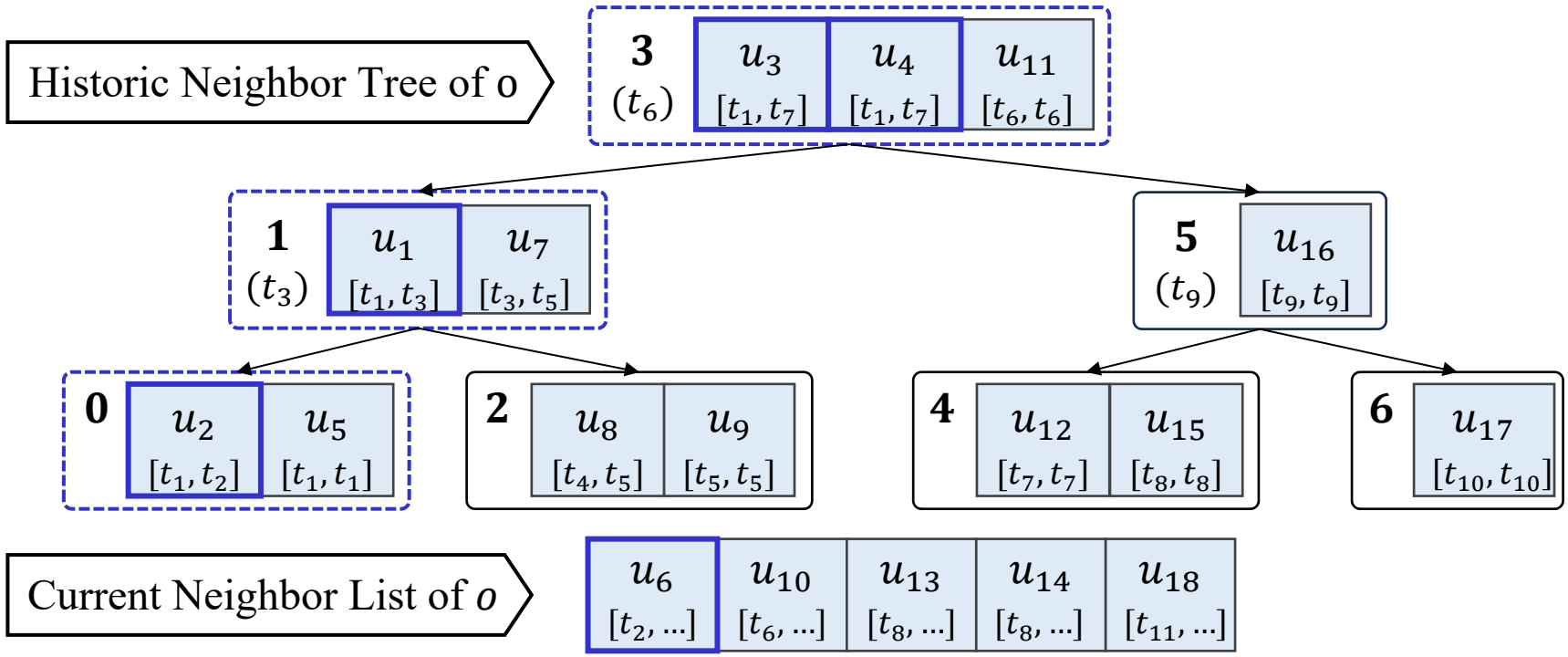
Compressed Timestamp Graph

- Design **historic neighbor tree** to store the neighbor list for points
  - Store each neighbor for only one time
  - Support efficient reconstruction of neighbor list for each timestamp



Compressed Timestamp Graph

- Design **historic neighbor tree** to store the neighbor list for points
  - For example: reconstruct neighbor list of o at t_2



Complexity Analysis

	Search	Update	Space
Original HNSW	$O(M\log N)$	$O(M\log N)$	$O(MN)$
Naïve Graph-based TANNS	$O(M\log N)$	$O(MN\log N)$	$O(MN^2)$
Timestamp Graph	$O(\log^2 N)$	$O(\log^2 N)$	$O(M^2N)$
Compressed Timestamp Graph	$O(\log^2 N)$	$O(\log^2 N)$	$O(MN)$

Same as the Original HNSW index

N: vector number in the dataset
M: neighbor number in the graph index

Outline

- Background and Motivation
- Problem Statement
- Method
- Experiment
- Conclusion

Experiment: Setup

- Datasets
  - Standard high-dimensional vector datasets [1] with randomly generated start & end timestamps
    - SIFT (128 dimensions, euclidean distance)
    - GIST (960 dimensions, euclidean distance)
    - DEEP (96 dimensions, euclidean distance)
    - GloVe (200 dimensions, cosine distance)
  - Real-world vector datasets with timestamps
    - TemporalWiki [2]: Wikipedia entities with time period, embedded with Longformer and DeBERTa

[1] ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. IS 2020.
[2] TemporalWiki: A Lifelong Benchmark for Training and Evaluating Ever-Evolving Language Models. EMNLP 2022.

Experiment: Setup

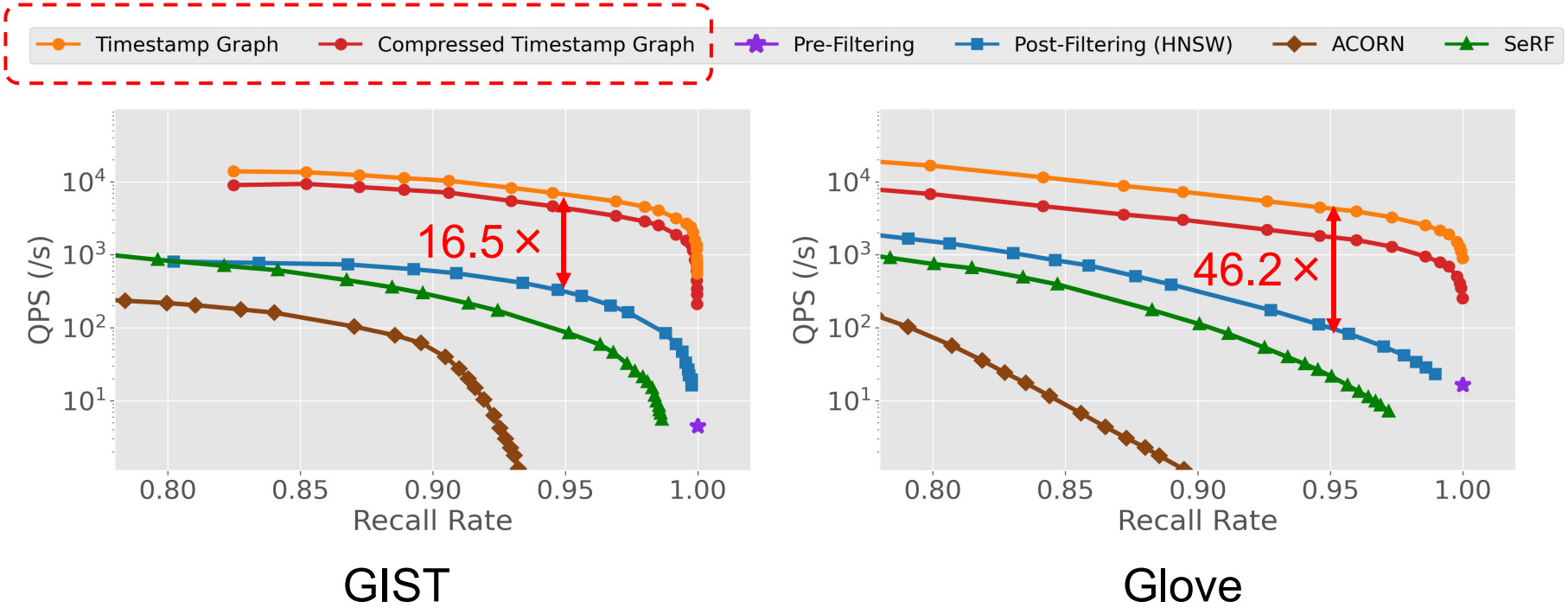
- Compared Method
  - Pre-Filter
  - Post-Filter (with HNSW)
  - SeRF[1]: method for range-filtering ANNS
  - ACORN[2]: predicate-agnostic method for filtered ANNS
  - Timestamp Graph: our method
  - Compressed Timestamp Graph: our method
- Metrics for Search Performance
  - Recall rate: $\frac{|r \cap r^*|}{k}$ (r: returned result, r^*: ground truth)
  - QPS: number of queries executed per second

[1] SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. SIGMOD 2024.
[2] ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. SIGMOD 2024.

Experiment: Result

- Search Performance
  - Accuracy: Over 99% recall rate for all datasets
  - Efficiency: 4.4× to 138.1× improvement in QPS compared to baselines at 95% recall

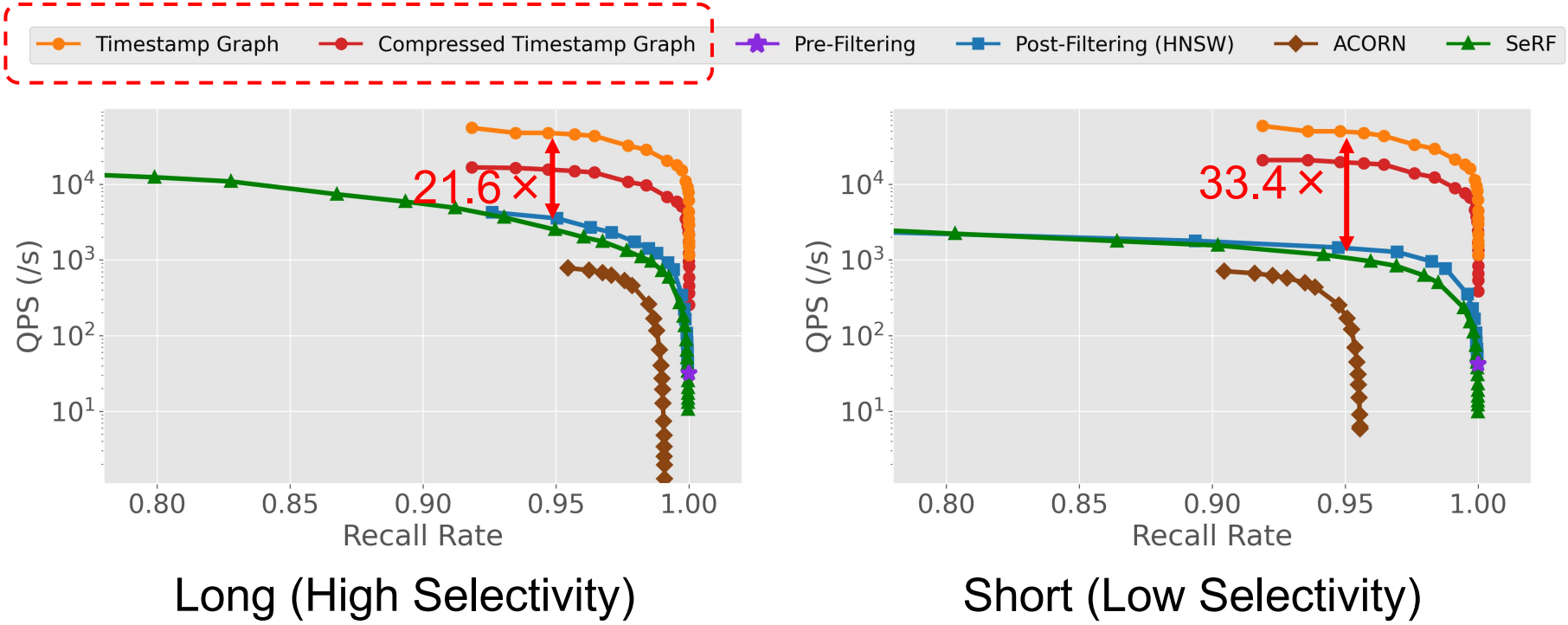
ours



Experiment: Result

- Search Performance
  - Varying Data Pattern: data pattern (i.e., selectivity) has small impact on our method

ours

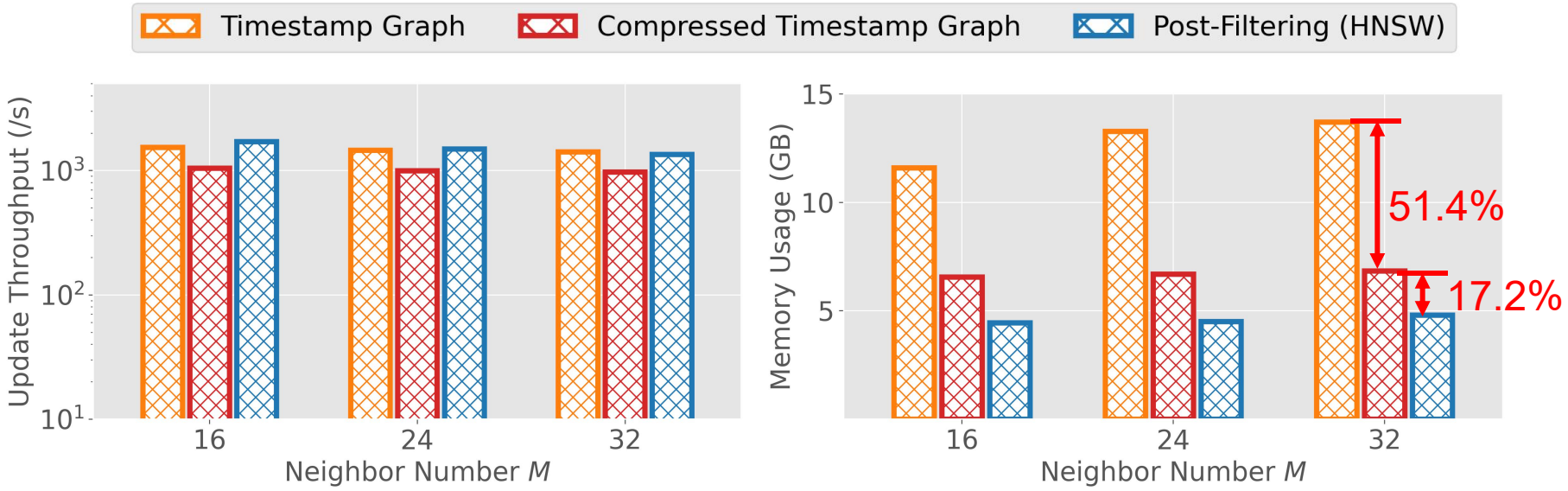


Long (High Selectivity)

Short (Low Selectivity)

Experiment: Result

- Index Construction
  - Update Efficiency: ranges from 0.8× and 1.5× of the original HNSW update throughput
  - Memory: compressed timestamp graph reduces memory cost by up to 51.4%



Outline

- Background and Motivation
- Problem Statement
- Method
- Experiment
- Conclusion

Conclusion

- We investigate TANNS, a new query in vector databases for emerging AI applications.
- We propose the timestamp graph to manage valid vectors across timestamps by a single index.
- We design the historic neighbor tree to compress the timestamp graph and achieve the same space complexity as HNSW index without timestamp information.
- Our solution yields a recall rate of over 99% on all datasets, while improving the query efficiency by 4.4× to 138.1× over the baselines.

Q & A

Thank You