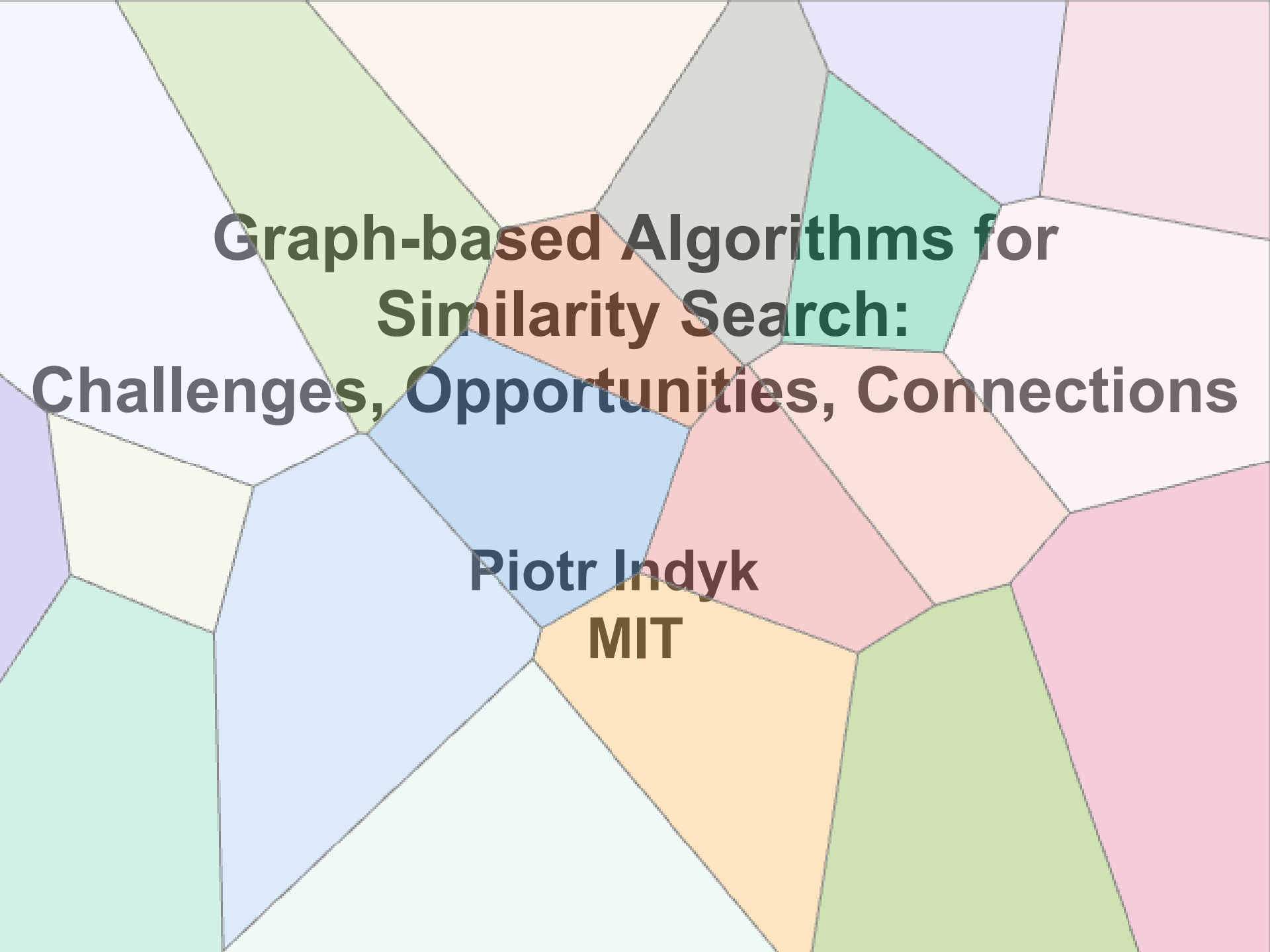


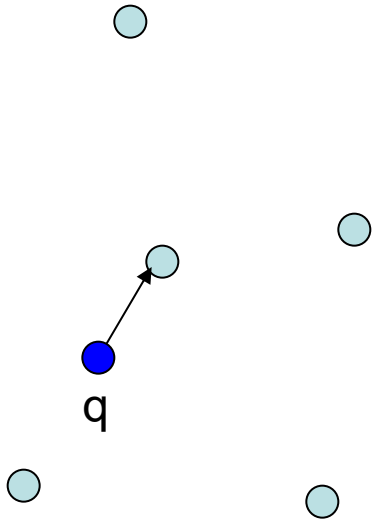
Graph-based Algorithms for Similarity Search: Challenges, Opportunities, Connections

Piotr Indyk
MIT

Similarity search

Nearest Neighbor Search

- **Given:**
  - a set P of n points
  - a metric $d(p,q)$ defining the dissimilarity between the objects p and q
- **Goal:** build a data structure which, given any query point q, returns $p \in P$ minimizing the dissimilarity $d(p,q)$
  - Often want k such closest points

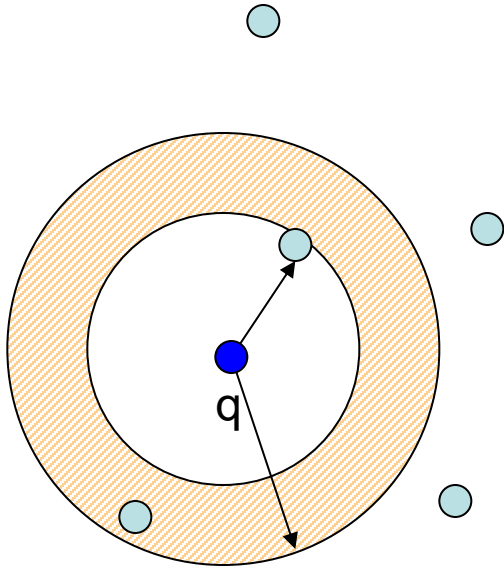


Relaxation I: Theory

- **Given:** a set P of n points, under some metric d, parameter $\varepsilon>0$
- **Goal:** data structure which, given any query q returns $p' \in P$, where

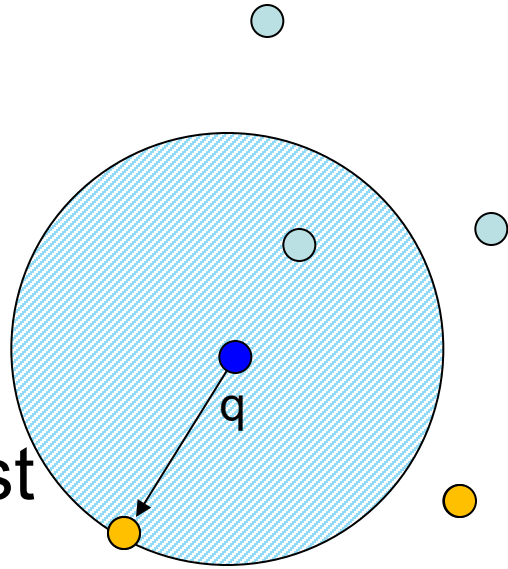
$$d(p',q) \le (1+\varepsilon) \min_{p \in P} d(p,q)$$

- Very convenient in theory
  - Connects to metric embeddings, randomized dimensionality reduction, etc



Relaxation II: Practice

- **Given:** a set P of n points, under some metric d, parameter k
- **Goal:** data structure which returns as many top k nearest neighbors as possible
  - Recall@k: the fraction of top k nearest neighbors returns



k=2

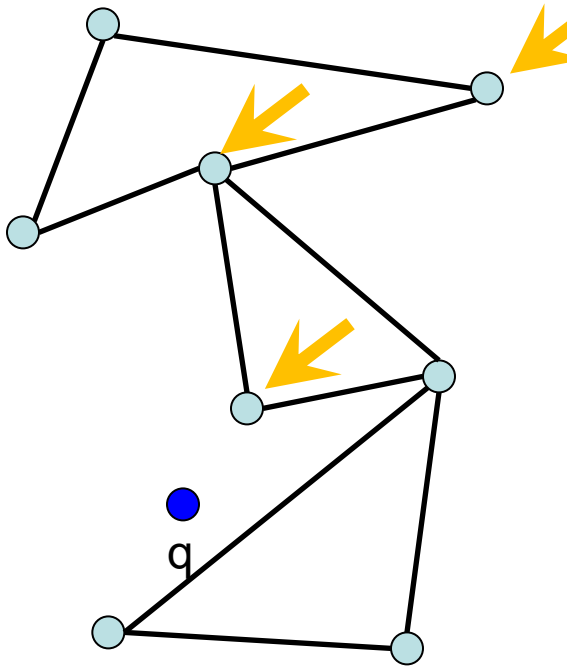
Recall@k=0.50

- These two relaxations are correlated, but distinct
- **In this talk**: we will use both, depending on the context

Graph-based algorithms

Graph-based algorithms

- **Preprocessing**: create a graph over points P (somehow)
- **Query answering:** greedy search
  - Generalized version uses bounded priority queue of size L
- Ideas go back to:
  - Orchard'91 (complete graph)
  - Arya-Mount'93 (for the Euclidean space)
  - Navarro'02 (for exact search)
  - Krauthgammer-Lee'04 (not quite greedy search)
  - See Clarkson'06 for a survey
- Last decade: HNSW, NSG, DiskANN, NGT, SSG, Kgraph, DPG, NSW, SPTAG-KDT, EFANNA ….



Three themes of this talk

1. Challenges: what can we say about the performance and correctness of those algorithms?

2. Opportunities: what else can we do with those algorithms ?

3. Connections: to LSH, quantization, etc

Part I: Correctness/performance guarantees

with Haike Xu

How this project has started

From: Harsha Vardhan Simhadri harshasi@microsoft.com

Piotr, Nice to meet you and thank you for invitation.

….

Re. the content, I wanted to share some empirical results on ANN design that underly our designs for large-scale “semantic/vector search indices” covering most Microsoft search and recommendation scenarios. I realize that yours is primarily a theory group, so I apologize in advance that I don’t have any theorems to share – our hope is that the discussion can help us bridge the gap between theory and practice.

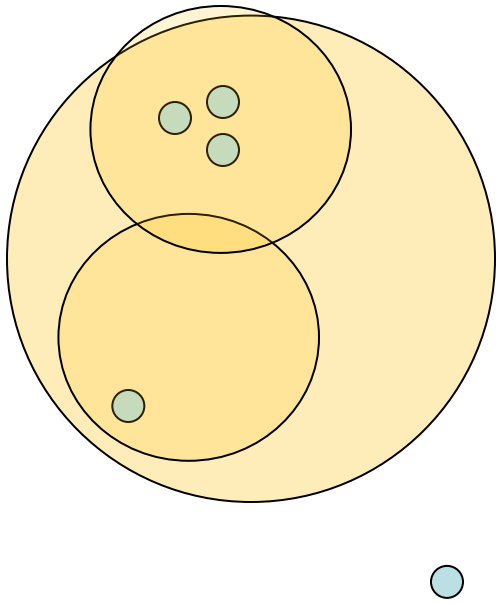
What is known about provable search algorithms in general metrics?

Authors	Space	Query Time
Krauthgamer, Lee'04	$2^{O(dim)}$ n log Δ	$2^{O(dim)}$ log Δ
Krauthgamer, Lee'04	n^2	$2^{O(dim)}$ $\log^2$ n
Har-Peled, Mendel'05	$2^{O(dim)}$ n log n	$2^{O(dim)}$ log n
Beygelzimer, Kakade, Langford'06	n	$2^{O(dim)}$ log Δ
Cole, Gottlieb'06	n	$2^{O(dim)}$ log n

Constant approximation factor; bounds up to O(.); Δ =diameter/closest-pair

Quantifying intrinsic dimension: doubling constant

- Consider a general metric $M=(X,d)$
- A **doubling constant** of M is the smallest value A such that **any** ball $B(p,2r)$ can be covered using at most A balls $B(p_1,r)\ldots B(p_A,r)$
  - $\dim=\log A$ is called **doubling dimension**



Past results

Authors	Space	Query Time
Krauthgamer, Lee'04	$2^{O(dim)}\, n \log \Delta$	$2^{O(dim)} \log \Delta$
Krauthgamer, Lee'04	n^2	$2^{O(dim)} \log^2 n$
Har-Peled, Mendel'05	$2^{O(dim)}\, n \log n$	$2^{O(dim)} \log n$
Beygelzimer, Kakade, Langford'06	n	$2^{O(dim)} \log \Delta$
Cole, Gottlieb'06	n	$2^{O(dim)} \log n$

Constant approximation factor; bounds up to O(.); Δ =diameter/closest-pair

Can we obtain similar approximation/performance guarantees for popular graph-based algorithms ?

Indyk, Xu'23: DiskANN (simplified preprocessing)	$2^{O(dim)}\, n \log \Delta$	$2^{O(dim)} \log^2 \Delta$

DiskANN

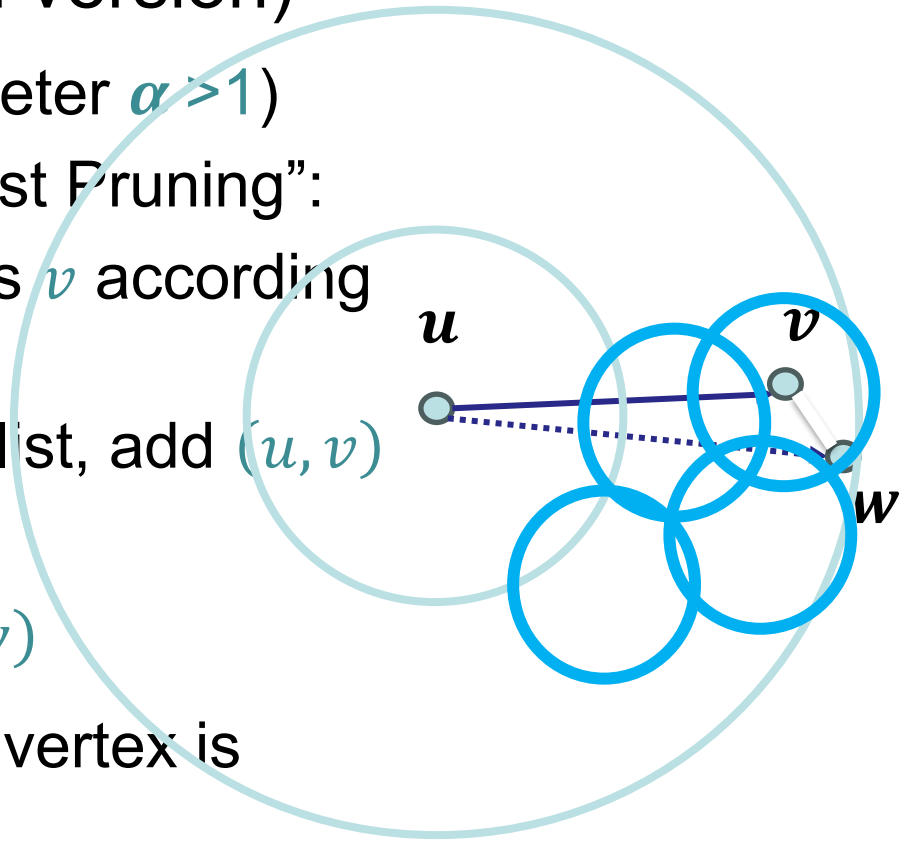
(simplified version)

Building the Graph (with parameter $\alpha > 1$)

For each point u, perform "Robust Pruning":

- Create a sorted list of all points v according to $d(u, v)$
- For each point v in the sorted list, add (u, v) and prune all w such that

$$d(v, w) \leq \frac{1}{\alpha} \cdot d(u, w)$$



Space: The out degree of each vertex is $\leq (4\alpha)^{dim} \cdot log\, \Delta$

Greedy search: yields a $\left(\frac{\alpha+1}{\alpha-1} + \epsilon\right)$ – approx. solution in $\log_\alpha \frac{\Delta}{\alpha(1-\epsilon)}$ iterations

Results

- Good news: correctness/runtime guarantees exist for **one** variant of **one** graph-based algorithm:

Indyk, Xu'23: DiskANN (simplified preprocessing)	$2^{O(\dim)} n \log \Delta$	$2^{O(\dim)} \log^2 \Delta$

- Bad news: unable to obtain similar bounds for any other popular algorithm
  - We tried ten of them
- In fact, we constructed families of synthetic point-sets and queries in 2D for which:
  - Either queries take linear time, or
  - Recall = 0%
- Main open problem: **empirically fast** algorithm with **provable** guarantees ?

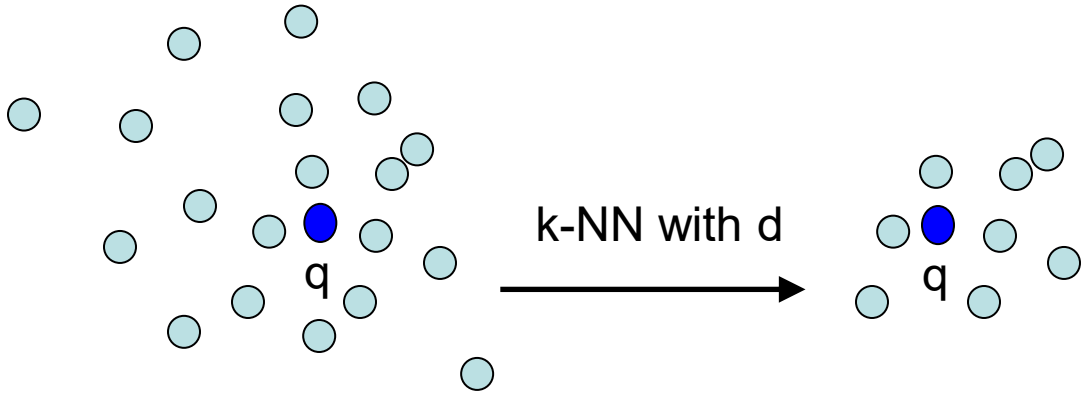
Some related work

- Guarantees:
  - S. Gollapudi, R. Krishnaswamy, K. Shiragur, and H. Wardhan. *Sort before you prune: Improved worst-case guarantees of the DiskANN family of graphs.* ICML, 2025.
- Performance:
  - S. Khanna, A. Padaki, and E. Waingarten. *Sparse navigable graphs for nearest neighbor search: Algorithms and hardness.* SODA, 2026.
  - A. Conway, L. Dhulipala, M. Farach-Colton, R. Johnson, B. Landrum, C. Musco, Y. Shechter, T. Suel, and R. Wen. *Efficiently constructing sparse navigable graphs.* SODA, 2026.
  - P. Avi, C. Musco, *Almost Navigable Graphs*, this workshop.
  - T. Rubel, R. Wen, G. Blelloch, L. Dhulipala, L. Gottesbüren, J. Łącki, V. Mirrokni. *Turbocharging PiPNN for Proximity and -NN Graph Building,* this workshop.

Part II: What else can we do with these algorithms ?

with Sandeep Silwal, Haike Xu

Filtering/Reranking



- Refining results using another (expensive but accurate) metric D. E.g.:
  - d induced by the Euclidean distance between embeddings, while D defined by a cross-encoder
  - d induced by a “sketch” of D
  - …
- Drawback: Accuracy limited by the first stage

Can we use graph-based nearest neighbor algorithms to reduce the drawbacks of filtering?

Bi-metric framework

- Assume:
  - Two metrics: “ground truth” metric D and “proxy” metric d
  - The distortion between d and D is at most C, i.e., $d(x, y) \leq D(x, y) \leq C \; d(x, y)$
    - E.g., think C=3
- Note that standard filtering pipeline guarantees only C-approximation

Improving over filtering: theory

Informal Theorem [Xu-Silwal-Indyk'26]:

Given a "graph-based" $(1+\varepsilon)$-approximate NN algorithms over metric d with space $S(n,dim,\varepsilon)$ and query time $Q(n,dim,\varepsilon)$, if we:

- perform **preprocessing (i.e., build the graph) using d** and
- answer **queries using D** (and modify the algorithm slightly)

then we get $(1+\varepsilon)$-approximation NN w.r.t D with:

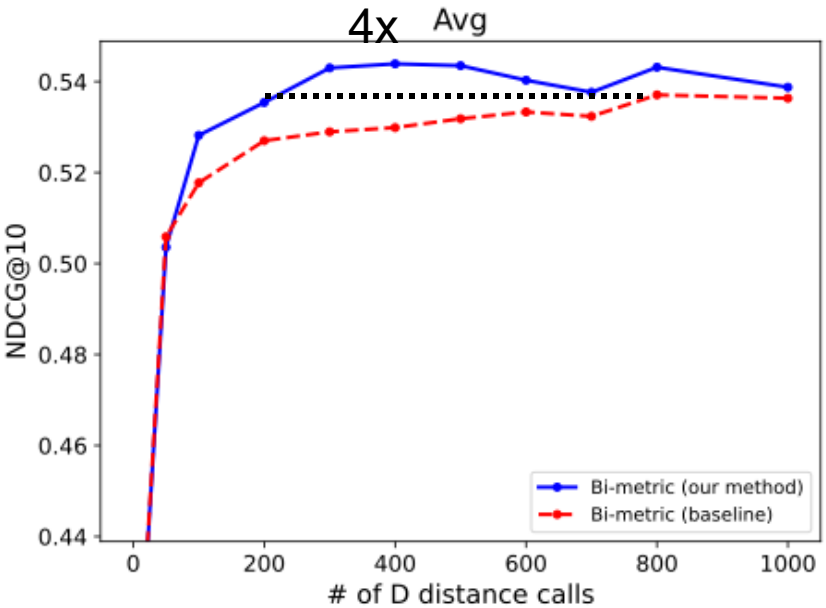
- space $S(n,dim,\varepsilon/C)$ and
- query procedure using $Q(n,dim,\varepsilon/C)$ calls to D

Improving over filtering: practice

- Text retrieval
- Experimented with DiskANN algorithm on MTEB benchmark data sets
- d=cheap embedding models, D=expensive cross-encoders
- Results: for several data sets, achieved state-of-the-art retrieval accuracy using up to **4x** fewer evaluations of expensive D compared to reranking

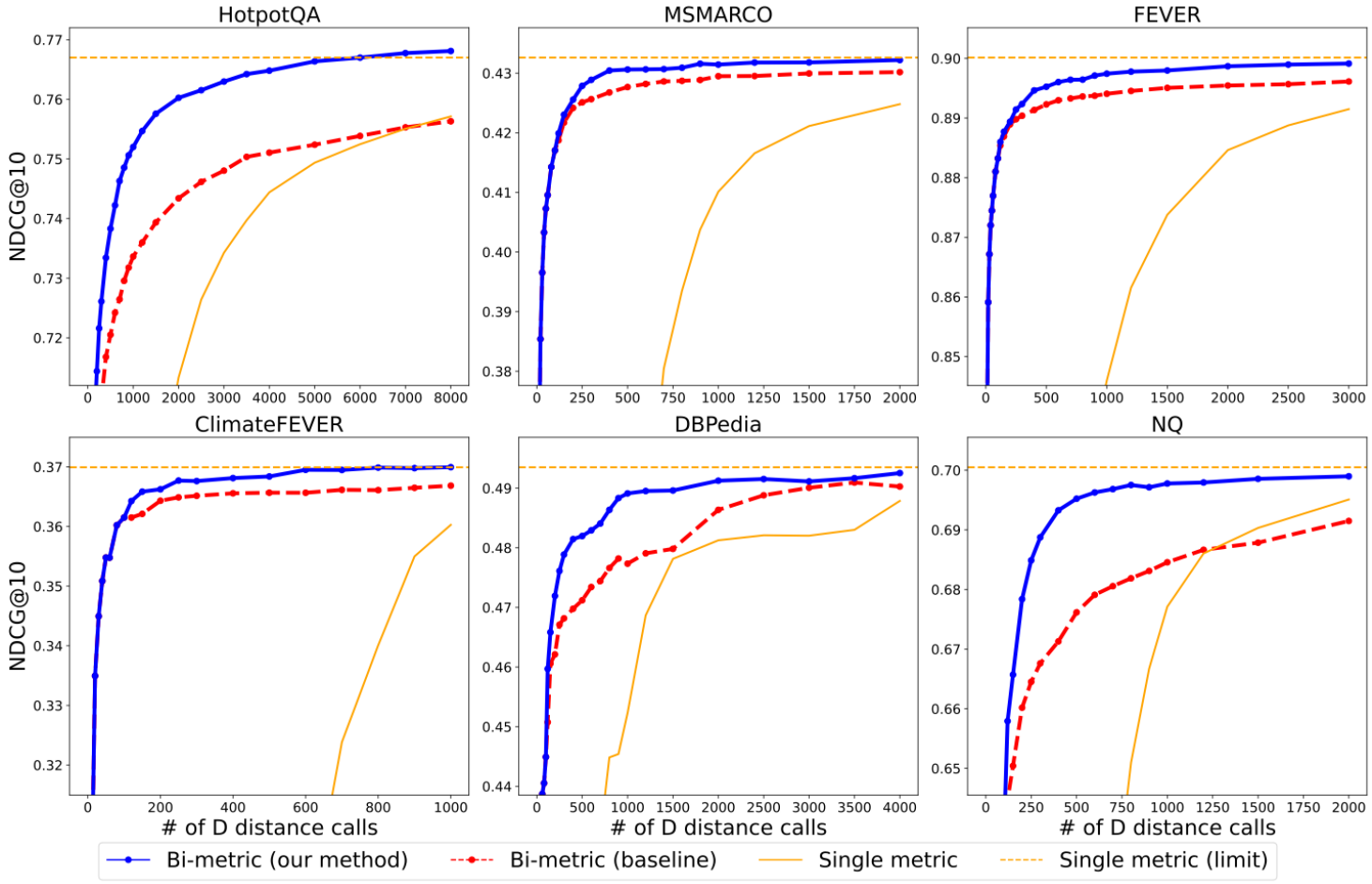
Results

- Setup:
  - Algorithm: modified DiskANN
  - d=bge-micro, D=Gemini 2.0 Flash
  - Data: large (>1 MB) MTEB benchmark data sets
    - Use only 500 queries, so that we can afford it
- Results (averaged over all data sets)



Results 2

d=bg-micro, D=SFR-Embedding-Mistral



Other work

- Haike Xu, and Tong Chen. *Beyond sequential reranking: Reranker-guided search improves reasoning intensive retrieval*. ICLR, 2026.
- P. Anand, P. Indyk, R. Krishnaswamy, S. Mahabadi, V. C. Raykar, K. Shiragur, and H. Xu. *Graph-based algorithms for diverse similarity search.* ICML, 2025.
- Much work at this workshop, e.g.,
  - M. D. Manohar, G. Blelloch, T. Kim. *Range Retrieval with Graph-Based Indices*
  - H. Xu, G. Blelloch, L. Dhulipala, L. Gottesbüren, R. Jayaram, J. Łącki. *JAG: Joint Attribute Graphs for Filtered Nearest Neighbor Search*
  - A. Amanbayev, B. Tsan, T. V. Dang, F. Rusu. *Filtered Approximate Nearest Neighbor Search in Vector Databases: System Design and Performance Analysis*

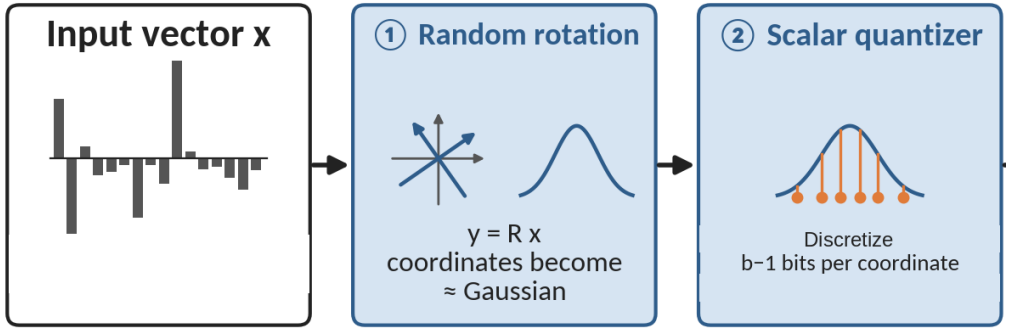
Part III: Connections



with Y. Feng, M. Kapralov, D. Krachun, B. Prokhorov

Where do proxy metrics come from ?

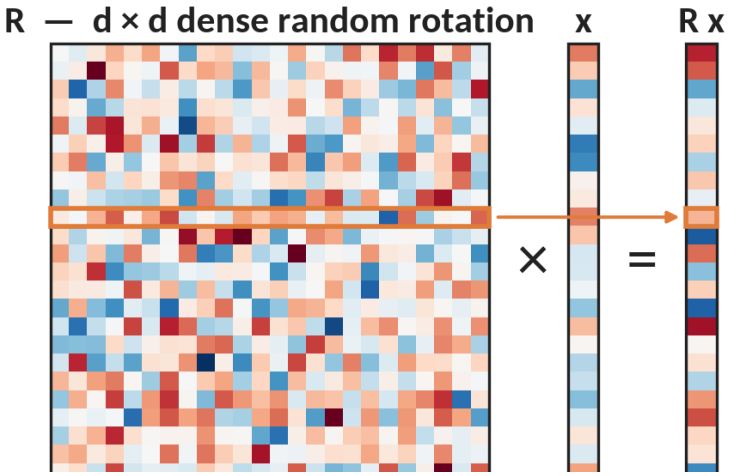
- A popular setting: d induced by a “sketch” of D
- For example:
  - D(p,q) is the Euclidean distance between p and q
  - d(p,q) is the Euclidean distance between quantized p and q
- Many quantization methods are known
- We focus on those based on random projections, e.g., Simhash, Eden, RabitQ, Turboquant



TurboQuant

A. Zandieh, M. Daliri, M. Hadian, V. Mirrokni, ICLR'26

- Components:
  - Uniformly random rotation R
  - Lloyd-Max quantizer optimized for Gaussian scalars
- Positives:
  - Expected MSE $\leq (\pi \sqrt{3}/2 + o(1)) \cdot 4^{-b}$
- Negatives:
  - Dense R, no structure ⇒ Rx costs $O(d^2)$ operations



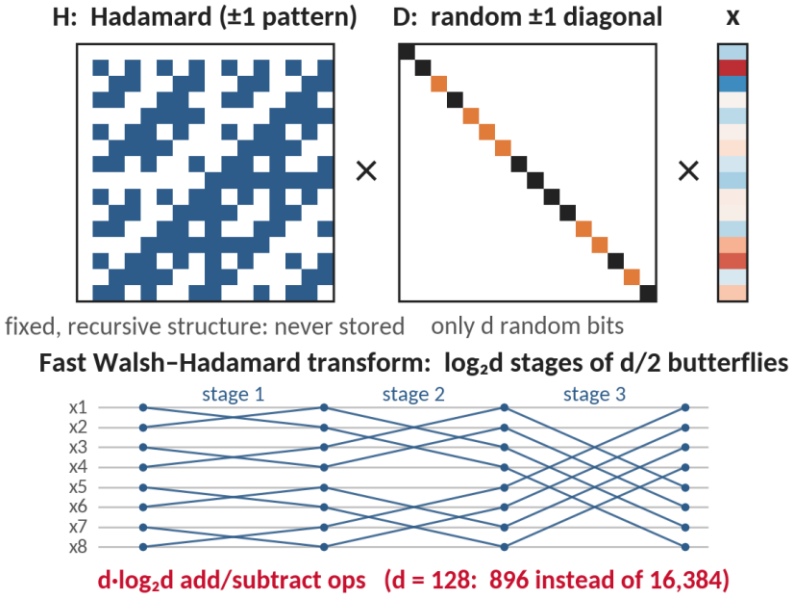
Our fix: randomized Hadamard + dithering

Feng, Indyk, Kapralov, Krachun, Prokhorov, 2026

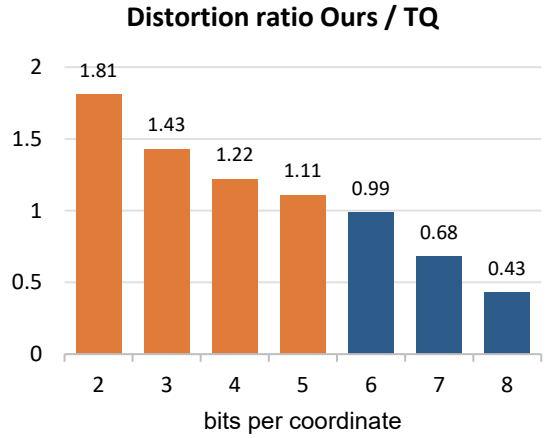
- Main ideas:
  - Replace R by HD: Hadamard × random ±1 diagonal ⇒ O(d log d) time
  - Dithering: random grid offset before rounding (the key to the guarantee)
- **Theorem:**

  MSE ≤ $(\pi\sqrt{3}/2 + o(1))\cdot 4^{-b}$

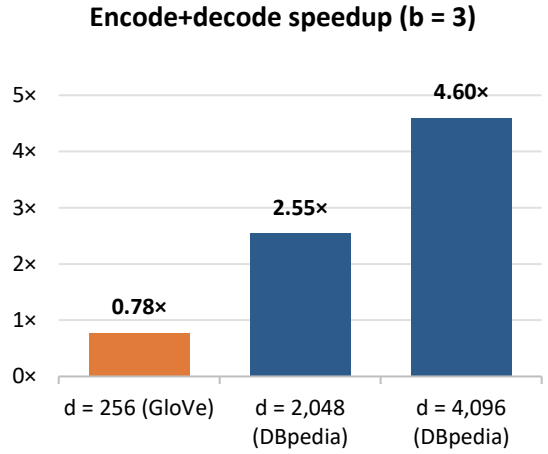
  - Same as before, but can't yet replicate finite b analysis



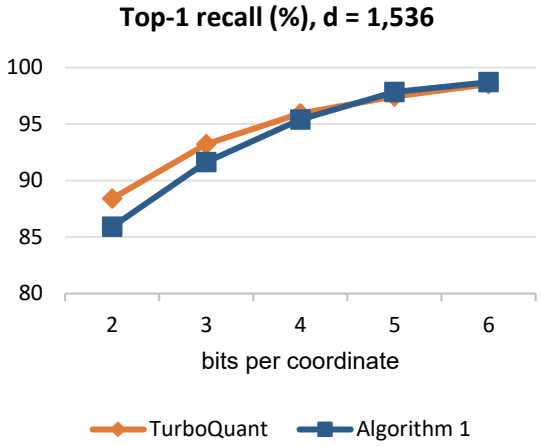
Experiments: TurboQuant vs. Ours



Higher error at low bits; our algo better from 6 bits up — same on all three datasets



Dense rotation cost grows with d^2: Our algo is slower only at d = 256, 4.6× faster at d = 4,096



→ Similar distortion, up to 4.6× faster
— the d^2 rotation was the bottleneck

Other work

- J. Gao, Y. Gou, Y. Xu, J. Shi, Y. Yang, S. Li, R. C.-W. Wong, C. Long. *Revisiting RaBitQ and TurboQuant: A Symmetric Comparison of Methods, Theory, and Experiments*, this workshop.

- R. Ben-Basat, W. Kuszmaul, M. Mitzenmacher, A. Portnoy, S. Vargaftik. *Quantizing With Randomized Hadamard Transforms: Efficient Heuristic Now Proven.* 2026.

- T. Rubel, R. Wen, G. Blelloch, L. Dhulipala, L. Gottesbüren, J. Łącki, V. Mirrokni. *Turbocharging PiPNN for Proximity and -NN Graph Building* (again).

Themes recap

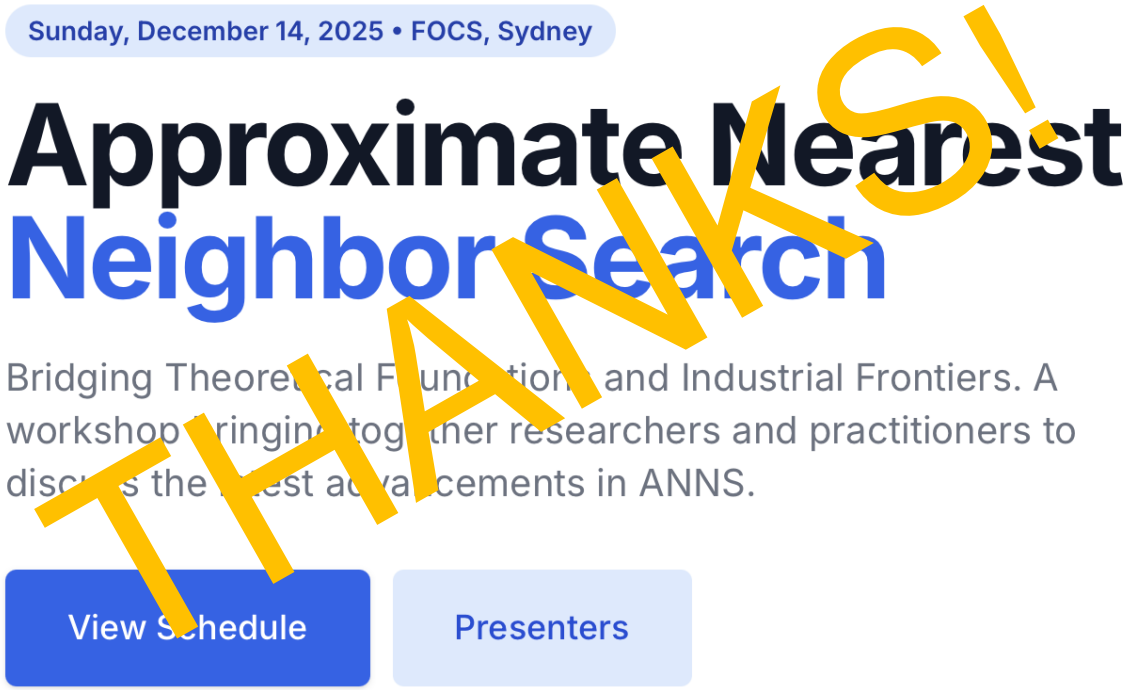
1. Challenges: what can we say about the performance and correctness of those algorithms?

2. Opportunities: what else can we do with those algorithms ?

3. Connections: to LSH, quantization, etc

FOCS 2025 workshop

(with Raj Jayaram, Ravi Krishnaswamy)



EXTRAS

Google

are there applications where lsh offers good results compared to hnsw

AI Mode | All | Short videos | Forums | Images | Shopping | Videos | More | Tools

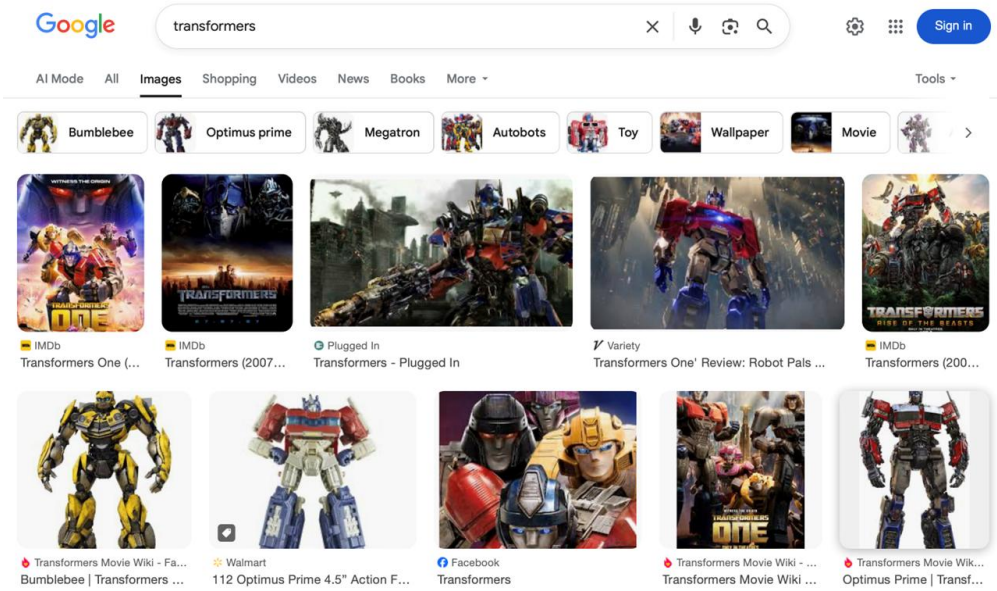
AI Overview

Yes, LSH excels over HNSW in specific scenarios like *extremely high-dimensional sparse data*, *near-duplicate detection*, and especially where *frequent, real-time index updates* (insertions/deletions) are needed, as HNSW struggles with graph maintenance and "unreachable nodes," while LSH offers simpler, faster updates to hash tables, though often trading recall for speed and memory efficiency compared to finely tuned HNSW.

Applications favoring LSH

- **Near-Duplicate Detection**: Minhash/Cosine LSH quickly finds very similar documents/images (e.g., plagiarism, content moderation).
- **High-Dimensional Sparse Data**: LSH can organize sparse vectors efficiently, a challenge for graph-based methods.
- **Frequent Data Updates**: Live feeds or dynamic datasets benefit from LSH's near-constant time insertion into hash tables, avoiding HNSW rebuilds.
- **Memory-Constrained Environments**: LSH can offer compact representations and lower memory overhead, ideal for edge devices or specific database features.
- **Audio/Music Fingerprinting**: Speed and scalability in audio recognition, though often using specialized LSH variants.

Diversity-aware search

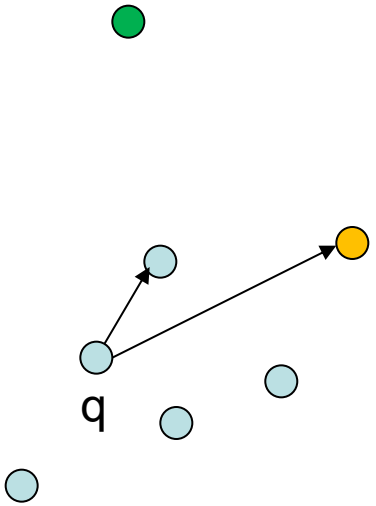


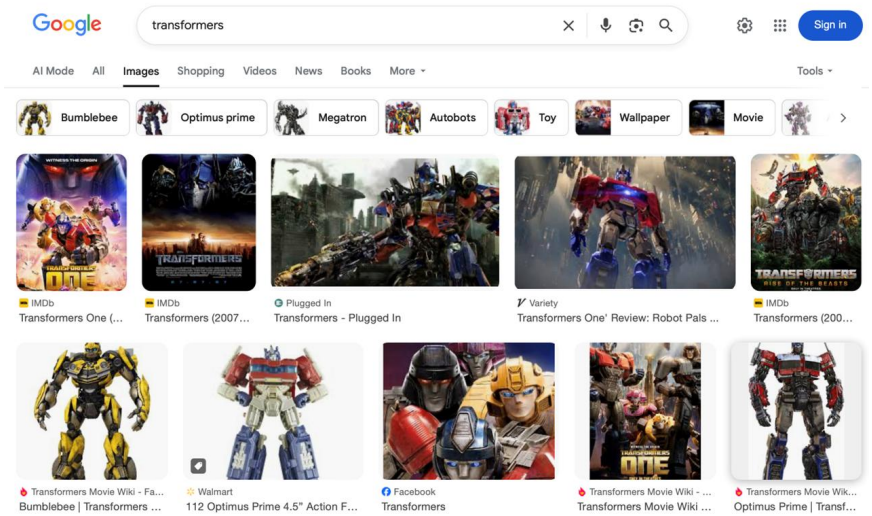
with P. Anand, R. Krishnaswamy,

S. Mahabadi, Raykar, K. Shiragur, H. Xu

Diverse Nearest Neighbor Search

- **Given:** a set P of n points in metric space , with colors and a parameter k
- **Goal:** build a data structure which, given any query q returns k points of different color minimizing the distance to q
- Motivation?





Prior work and our results

Authors	Comment	Space	Query Time
Abbar, Amer-Yahia, Indyk, Mahabadi, Varadarajan'13	d=Hamming approx=O(c)	log k $n^{1+1/c}$	k^2 log k $n^{1/c}$

We were able to modify existing non-diverse DiskANN graph-based algorithm to give the first graph-based algorithms for the diverse problem.

- Space: multiplied by k
- Time: multiplied by k^2 (or k for the colorful version)

Diverse DiskANN

Building the Graph:

- Pruning
  - If $d(v,w) \leq \frac{1}{\alpha} \cdot d(u,w)$
    - Prune the edge (u,w) only if either
      1. $col[v] = col[w]$
      2. Or we have connected u to at least k different colors in the ball of radius $\frac{1}{\alpha} \cdot d(u,w)$ around w

Diverse DiskANN

- **Query answering algorithm:**
  - Start from k points that all have different colors $a_1, \dots, a_k$
  - In each iteration, **swap one point** a_i with a neighbor point a' that
    - is closer to the query
    - has a different color from the rest of the points

Experiments

Algorithms:

Baseline: Standard DiskANN + Postprocessing to ensure diversity

Our algorithm 1: Standard DiskANN Build + Diverse DiskANN Search

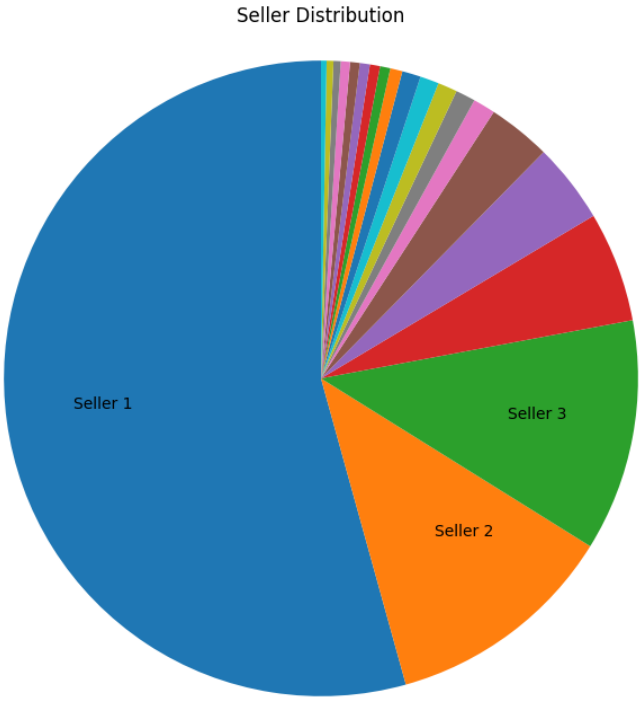
Our algorithm 2: Diverse DiskANN Build + Diverse DiskANN Search

Datasets:

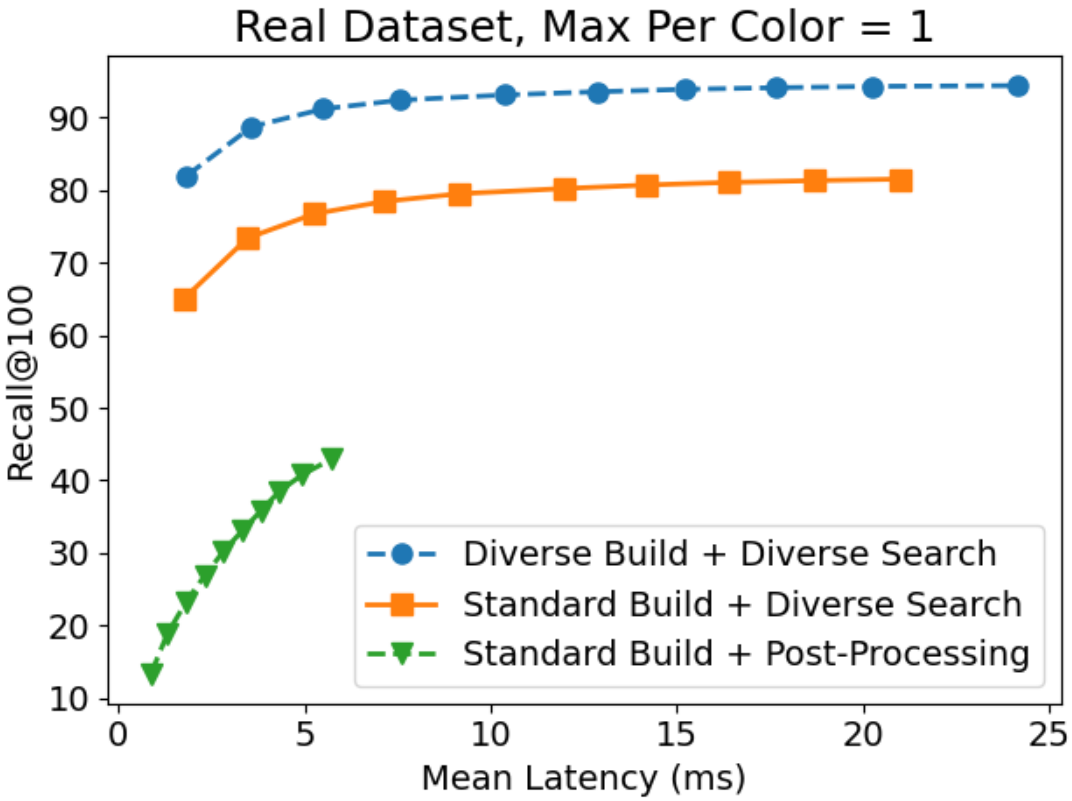
- **Ads dataset**: 20 Million vectors, 5000 queries, 64 dimensions
- **Also: Semi-Synthetic Arxiv**: **Semi-Synthetic SIFT**

Parameters:

- $k = 100$



Experiments: Recall vs Latency

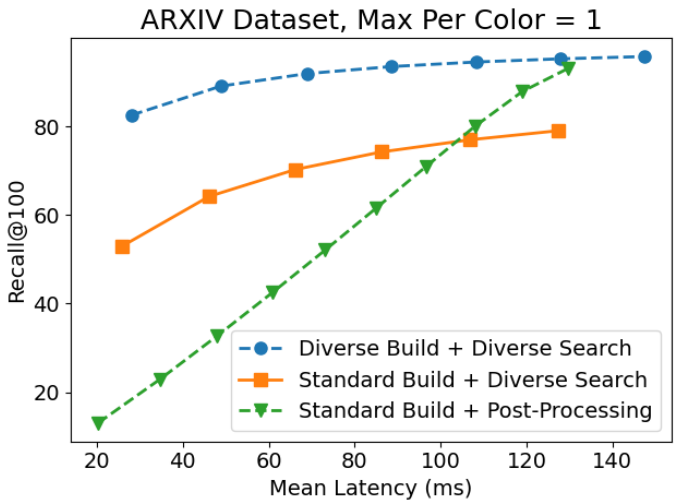


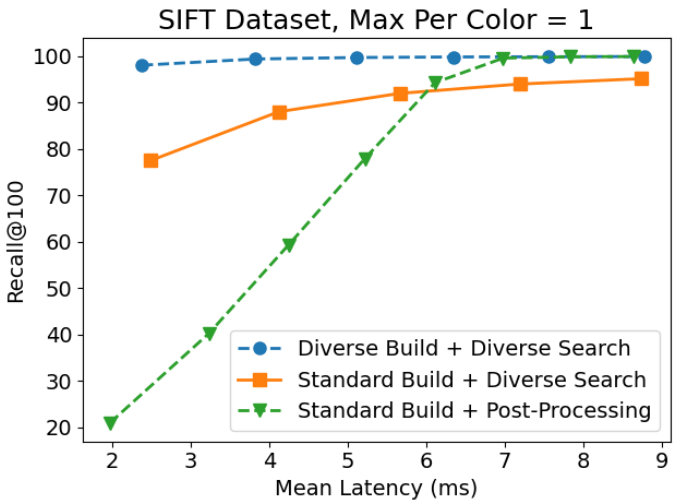
Baseline: Standard DiskANN + Postprocessing to ensure diversity

Our algo 1: Standard DiskANN Build + Diverse DiskANN Search

Our algo 2: Diverse DiskANN Build + Diverse DiskANN Search

Experiments ctd





Baseline: Standard DiskANN + Postprocessing to ensure diversity

Our algo 1: Standard DiskANN Build + Diverse DiskANN Search

Our algo 2: Diverse DiskANN Build + Diverse DiskANN Search

Summary

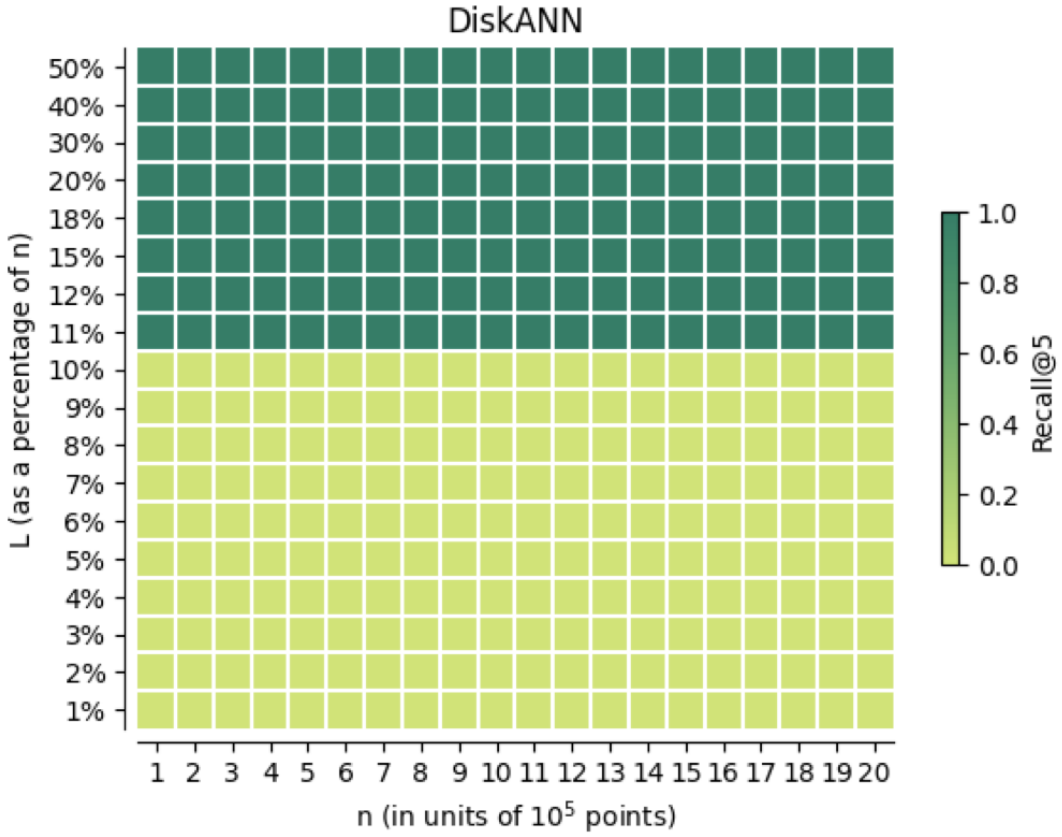
- Extended DiskANN to diverse nearest neighbor
  - Space: multiplied by k. Time: multiplied by k^2 (or k for colorful variant).
- Questions:
  - Better space/time bounds?

  Recent result by Samson Zhou's group for the colorful version:
    - Space: multiplied by log k. Time: multiplied by k. (but randomized)
    - Constant space overhead?

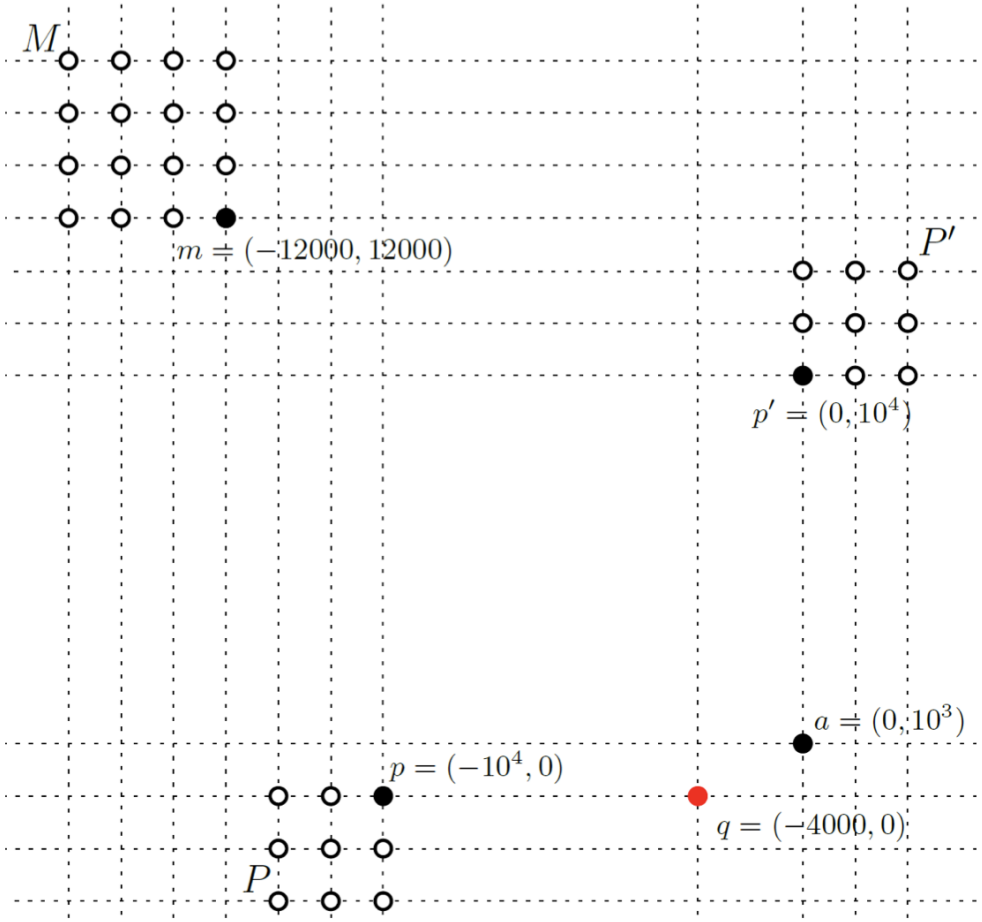
“Fast” DiskANN: Preprocessing

- Initialize the graph to a random R-out graph
- Repeat twice
  - For each $p \in P$
    - Perform greedy search for the nearest neighbor of p
    - Connect p to/from the vertices scanned by greedy search
    - Perform **Robust Pruning** on any node that has $>R$ neighbors, keeping at most R of them
- Does this offer provable guarantees?

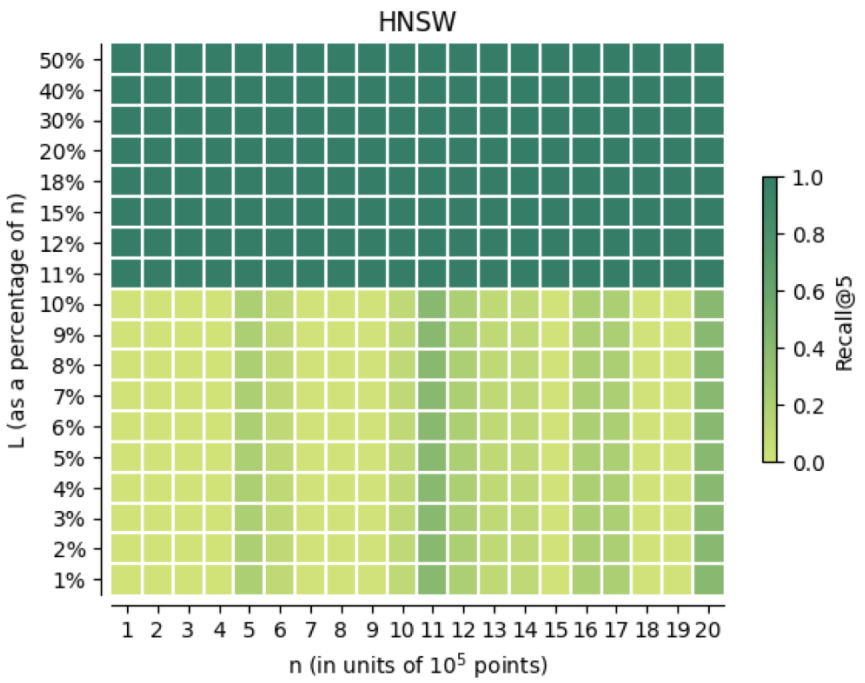
DiskANN with fast preprocessing: results

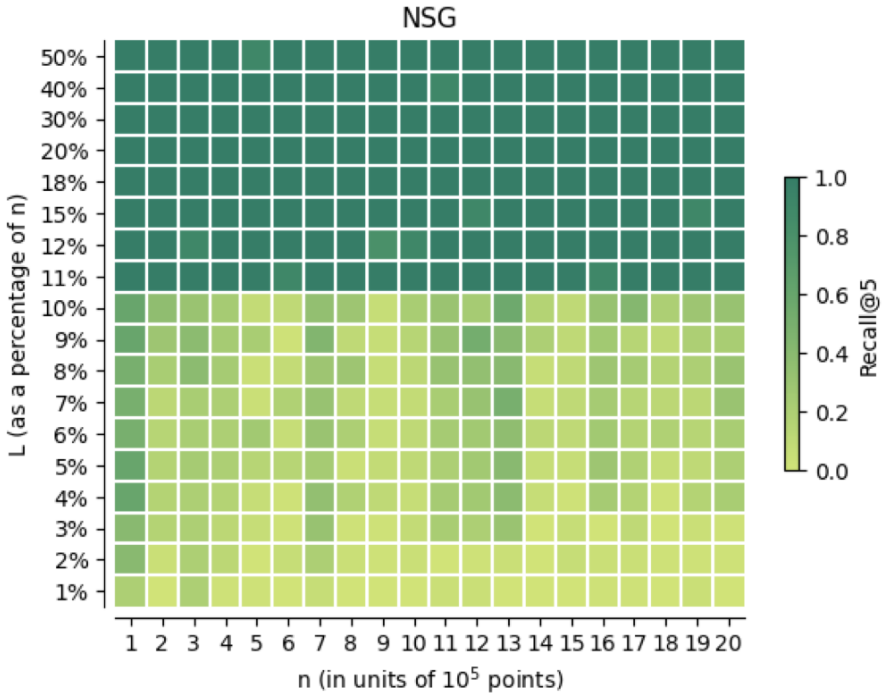


Hard data set for DiskANN



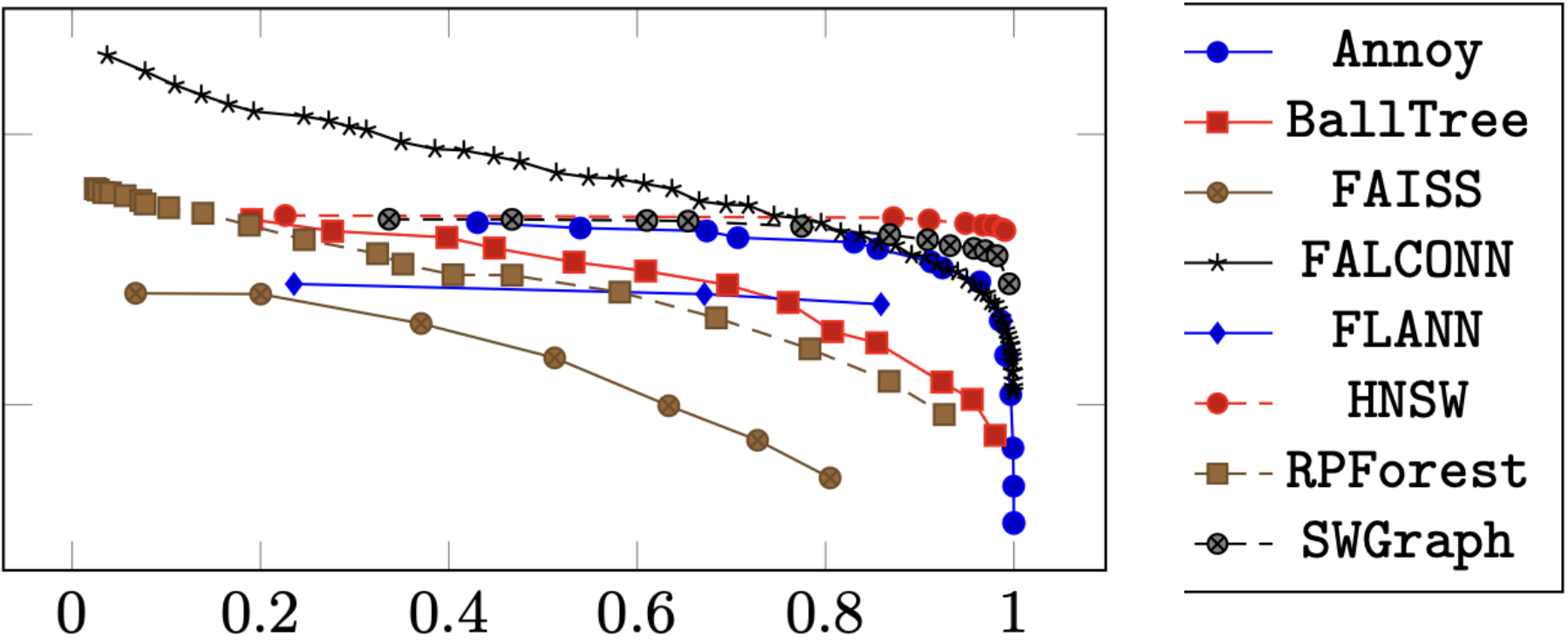
HNSW, NSG



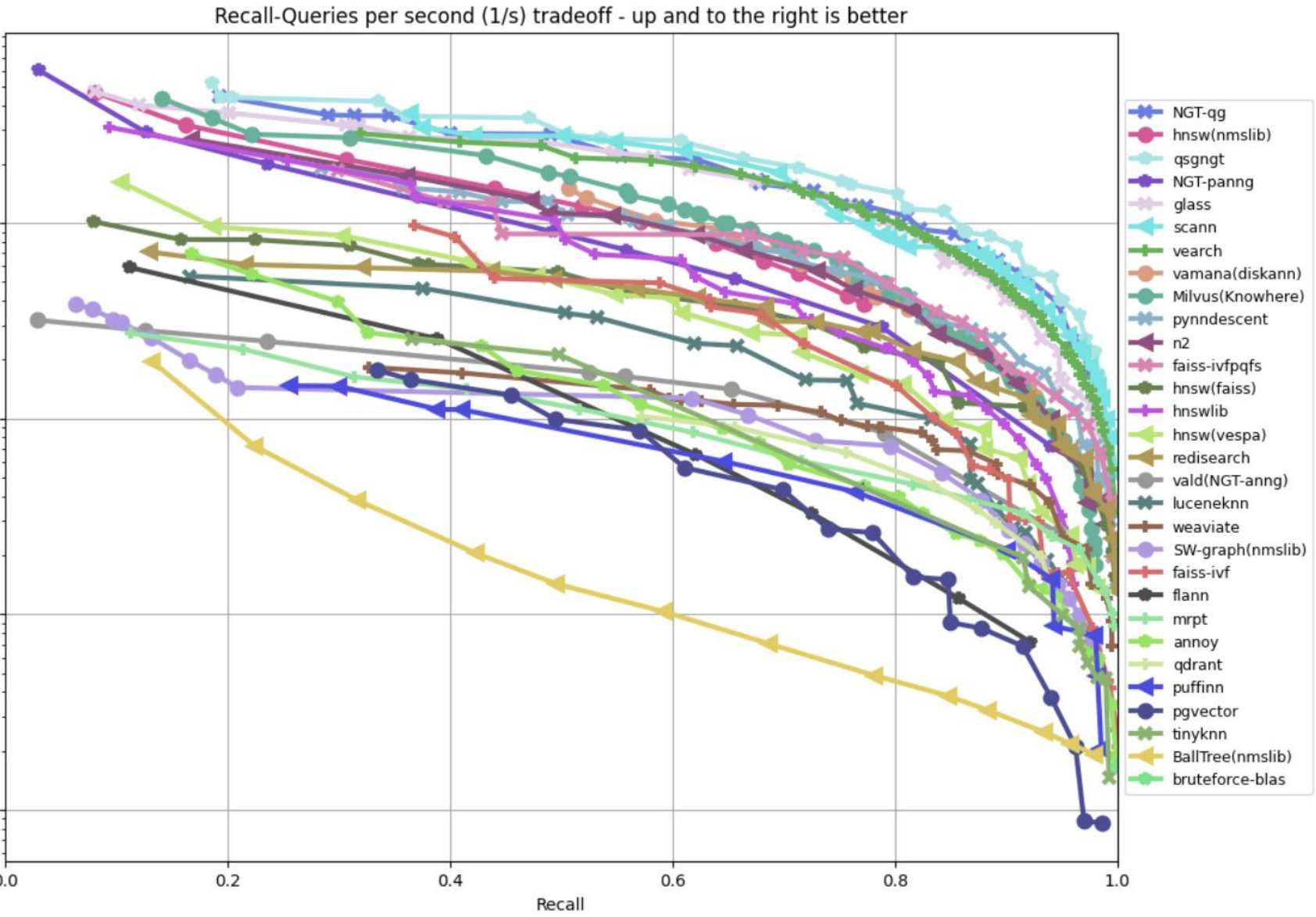


- Similar results hold for NGT, SSG, Kgraph, DPG, NSW, SPTAG-KDT, EFANNA

Landscape in 2017



From: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. SISAP, 2017.



From: https://ann-benchmarks.com