Almost Navigable Graphs

Pratyush Avi, Christopher Musco
New York University

Let $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^r$ be a set of vectors and let $d : \mathbb{R}^r \times \mathbb{R}^r \mapsto \mathbb{R}$ be some distance function between vectors.

Nearest Neighbor Search:
For any query vector $q \in \mathbb{R}^r$, we want to find a vector $x^* \in P$ such that

$$x^* = \text{argmin}_{x \in P}\, d(q, x)$$

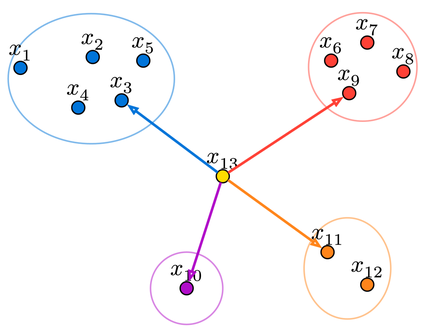
Graph-Based Vector Search:
Pre-process the dataset into a graph on which we can perform nearest neighbor search. Here each point in the dataset is a vertex in the graph.

Popular examples: Microsoft DiskANN, NSG, HNSW

Navigable Graphs

Definition (Navigbaility): A graph $G = (P, E)$ is navigable if, for all $i, j \in [n]$, there is an edge $(x_i, u) \in E$ such that $d(u, x_j) < d(x_i, x_j)$.

Greedy search on navigable graphs always succeeds on *in-distribution* queries, i.e., greedy search for any query $x_j \in P$ will always return x_j for any start point.



Existing theory about navigability:
- $\Omega(n^{1.5})$ edges in the worst case [1]
- $\Omega(n^2)$ construction time [2,3]

[1]Diwan, Gou, Musco, Musco, and Suel 2024
[2]Conway, Dhulipala, Farach-Colton, Johnson, Landrum, Musco, Shechter, Suel, and Wen 2026
[3]Khanna, Padaki, and Waingarten 2026

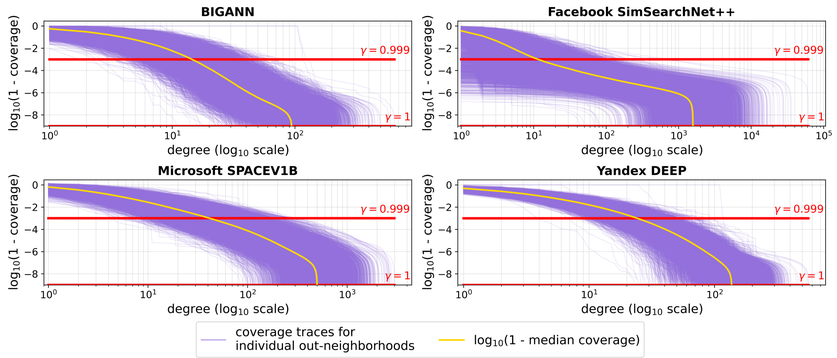
Are there relaxations of navigability that allow sparser graphs and faster construction time?

Most practical algorithms construct navigable graphs in some heuristic sense. We formally explore a natural relaxation of navigability.

γ-Almost Navigable Graphs

Definition (γ-Almost Navigbaility): A graph $G = (P, E)$ is γ-almost navigable if every $x_i \in P$ has an edge $(x_i, u) \in E$ such that $d(u, x_j) < d(x_i, x_j)$ for at least $\gamma \cdot (n-1)$ unique x_j.

True navigability requires each out-neighborhood cover $n-1$ points, while γ-almost navigability only requires coverage of $\geq \gamma(n-1)$ points.



On billion-scale benchmark datasets, we observed that it takes an ***order of magnitude*** more edges to achieve full navigability, as opposed to 99.9%-almost navigability. We have theory to support this:

Theorem: For any point set P and $\gamma \in [0,1)$, there exists a γ-almost navigable graph with average out-degree $O(1/(1-\gamma))$ that can be constructed in $\tilde{O}(n/(1-\gamma))$ time.

This implies, for constant γ we can obtain:
- $O(n)$ edge graphs
- in $\tilde{O}(n)$ construction time!

Constructing Almost Navigable Graphs

Algorithm Outline:

(1) Arbitrarily partition the dataset into subsets of $\lceil 2/(1-\gamma) \rceil$ points, $S_1, \ldots, S_k$.

(2) Add a clique to each S_i.

(3) Let $\bar{P}$ be the points with coverage $< \gamma(1-n)$.

(4) If $|\bar{P}| \geq \lceil 2/(1-\gamma) \rceil$, repeat from step 1 with points in $\bar{P}$.

(5) Else, connect every point in $\bar{P}$ to the entire dataset.

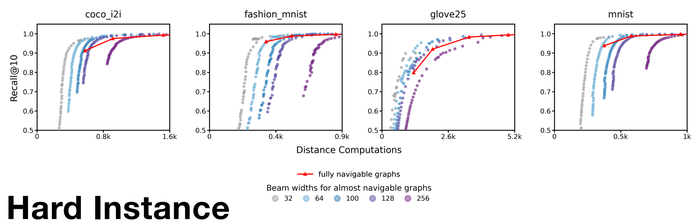
Randomizing step 3, yields a linear time algorithm.

Search Performance

We performed beam-search on almost navigable graphs for various choices of γ and beam-width.

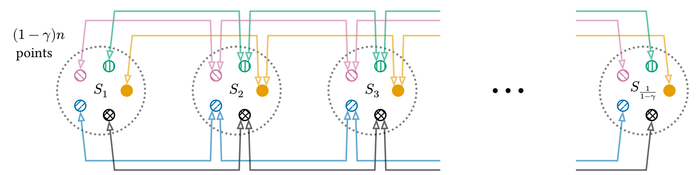
To achieve similar or better **recall@k** than fully navigable graphs, almost navigable graphs had

- **25-47%** lower search cost (distance computations)
- **46-55%** smaller graphs (vertex degree)



Hard Instance

Unfortunately, almost navigable graphs don't enjoy the same theoretical guarantees about search quality as fully navigable graphs.



Here, greedy search fails on all but $1/(1-\gamma)$ in-distribution search queries for any start point.