Sparse Navigable Graphs for Nearest Neighbor Search

Sanjeev Khanna

Univ. of Pennsylvania

Joint work with: Ashwin Padaki (Univ. of Pennsylvania)
Erik Waingarten (Univ. of Pennsylvania)

Nearest Neighbor Search (NNS)

Input: A dataset P containing n elements, an underlying distance metric d, and a query point q.
Goal: Find a point in P that is approximately-closest to q.

- Image/product search: find visually similar photos or items from a given example.
- Recommendations & personalization: retrieve nearest users/items from learned embeddings.
- Retrieval-Augmented Generation: LLMs fetch relevant documents or memories via nearest neighbors.

A New Approach: Graph-based NNS

NNS via greedy local exploration

- Each point in P stores edges to a few nearby points.
- An NNS query q is answered by a greedy walk to a neighbor closest to the target:
  - start at an initial point s, and
  - repeatedly move to a neighbor closer to the target.
- Dataset geometry is captured implicitly through local connectivity.
- Enables fast, scalable search in practice (DiskANN, HNSW).

Graph Structure and NNS Performance

- Graph NNS works well in practice when graphs have low degree and short greedy paths.
- Theoretical perspective: search time can be viewed roughly as (greedy path length) x (max vertex degree).

Sparse graph with short greedy paths $\Longrightarrow$ Fast NNS search.

Motivating Question: How can we construct sparse graphs that still guarantee efficient greedy navigation for NNS?

Navigable Graphs

- Given a dataset P with metric d, a directed graph $G(P, E)$ is navigable if for all $s \neq t \in P$, there is an edge $(s, u) \in E$ s.t.
$$d(u, t) < d(s, t).$$
- This definition ensures that given any target point $t \in P$, no matter the starting point, greedy search will end up at t.

Some issues …

- The search path may be very long.
- This requirement is too weak to recover a good answer when query point $q \notin P$.

α-Navigable Graphs: A Small-world fix

[Indyk-Xu '23]
Given a dataset P with metric d , a directed graph $G(P,E)$ is α-navigable if for all $s \neq t \in P$, there is an edge $(s,u) \in E$ s.t.

$$d(u,t) < d(s,t)/\alpha.$$

Idea: Each greedy step makes multiplicative progress.

Theorem [Indyk-Xu '23]
For any , if G is α-navigable then greedy search returns a $\left(\frac{\alpha+1}{\alpha-1}+\epsilon\right)$-ANN in $O(\log\left(\frac{\Delta}{\epsilon}\right))$ hops where Δ is the aspect ratio.

Takeaway: Sparse α-navigable graphs $\Longrightarrow$ Fast NNS search!

Building Sparse α-Navigable Graphs

Slow-DiskANN:

- Repeatedly add an edge from source s to nearest vertex in $P \setminus \{s\}$ whose α–navigability constraint is not covered.
- A natural strategy for building sparse α–navigable graphs.

[Indyk-Xu '23] Slow-DiskANN builds an α-navigable graph with $deg_{max} \leq (4\ \alpha)^{\lambda(P)}$ (here $\lambda(P)$: doubling dimension).

[Diwan et. al. '24] For any metric d, there is a 1-navigable graph with $deg_{avg} \leq \tilde{O}(\sqrt{n}\,)$.

Limitations of General Approaches

Result 1 [K, Padaki, Waingarten '25]
There is a dataset P for which there is an α-navigable graph G of max degree $O(\log n)$, but where Slow-DiskANN outputs a graph of max degree $\Theta(n)$.

- Slow-DiskANN gives $\tilde{\Omega}(n)$–approximation to max degree on this instance.

α-Sparsest Navigable Subgraph Problem

α-Sparsest Navigable Graph Problem (α-SNP)

Given a dataset (P, d) and $\alpha \geq 1$, what is the sparsest α-navigable graph on P?

- We will measure sparsity in terms of max degree but our results carry over to total number of edges.

Questions:

- What is the complexity of exact α-SNP ?
- How fast can we (approximately) solve it?

α-SNP and Set Cover

Result 2 [K, Padaki, Waingarten '25]
α-SNP is equivalent to the Set Cover problem.

- Inherits algorithms and hardness results for Set Cover.
- α-SNP is NP-hard to approximate to within a $\Theta(\ln n)$ factor.
- On the other hand, standard greedy set cover algorithm can be implemented in $O(mn)$ time when the instance contains m sets and n elements.
- This gives a baseline $O(n^3)$ time $O(\ln n)$-approximation algorithm for α-SNP.

α-SNP Reduces to Set Cover

Recall α-navigability from a source vertex s requires: for all $t \in P \setminus \{s\}$, there is an edge $(s, u) \in E$ s.t.

$$d(u, t) < d(s, t)/\alpha.$$

Equivalent Set Cover Instance

- Points in $P \setminus \{s\}$ are universe U of elements for set cover.
- For each $u \in P \setminus \{s\}$, define a set $Z(s, u) = \{\, t \mid d(u, t) < d(s, t)/\alpha\}$ (the family of sets F).
- So minimizing $\deg(s)$ $\Leftrightarrow$ minimizing set cover for (U, F).

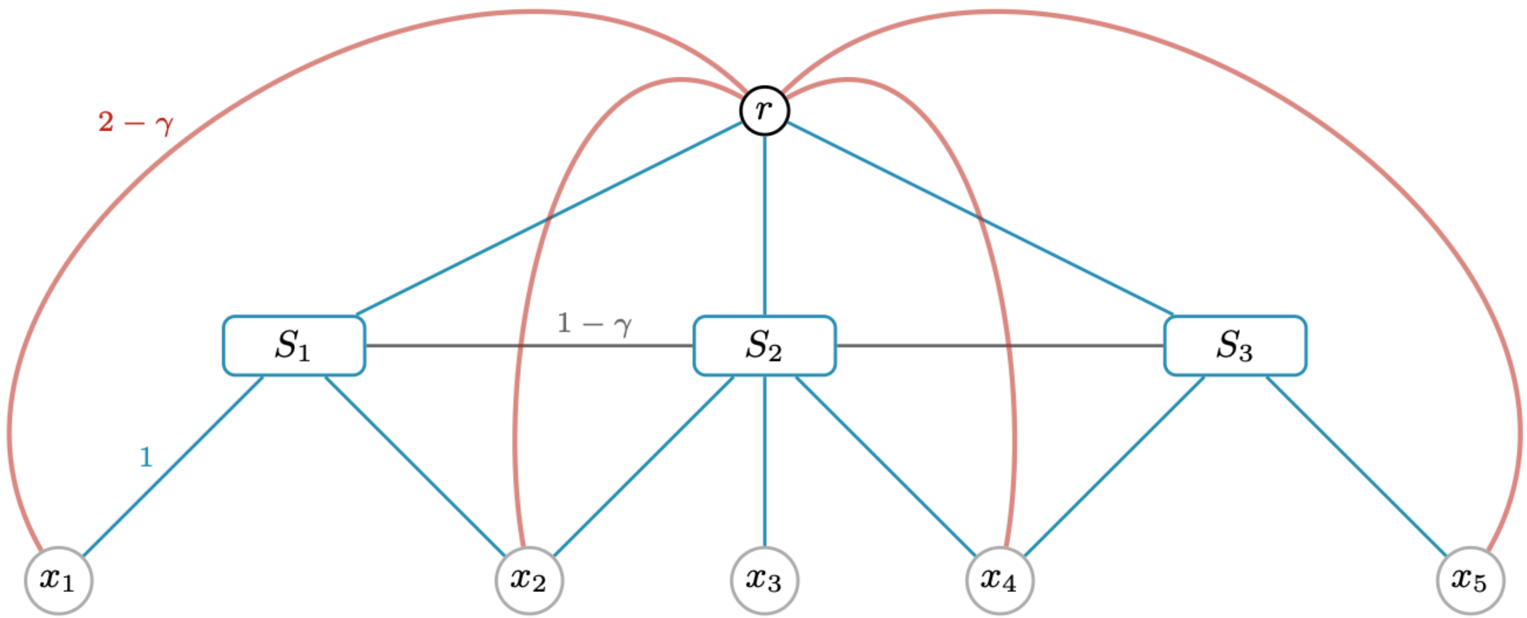
Overall, we solve n set cover instances: one for each $s \in P$.

Set Cover Reduces to α-SNP

Given a set cover instance (U, F), we build a dataset (P, d) such that optimal max degree of a 1-navigable graph on (P, d) is proportional to optimal set cover size on (U, F).

- Let $U = \{x_1, x_2, \ldots, x_n\}$, and $F = \{S_1, S_2, \ldots, S_m\}$.
- P contains a special root vertex r, as well as vertices representing elements in U and sets in F.
- Metric d ensures that navigability constraints from the root vertex r require that for each element $x_j \in U$:
  - either we have a direct edge from r to x_j, or
  - an edge from r to a set S_i such that $x_j \in S_i$.
- Minimizing out-degree from r $\Leftrightarrow$ minimizing set cover size.

Set Cover Reduces to Sparse Navigability



$d(r, S_i) = 1$, $d(r, x_j) = 2 - \gamma$, and $d(S_i, x_j) = 1$ if $x_j \in S_i$.

So $d(S_i, x_j) < d(r, x_j) \Leftrightarrow x_j \in S_i$.

Set Cover Reduces to Sparse Navigability

- The basic gadget is not sufficient as max degree will be dominated by set-set and element-element navigability constraints.

- Make many copies of this gadget to ensure that max degree is determined by the optimal set cover cost.

- In particular, $L = (m + n)^2$ copies of the basic gadget where each of the L roots is connected to sets/elements in each gadget ensures that max degree $\propto$ set cover cost.

Faster Algorithms via Set Cover Connection

Baseline solution for $O(\ln n)$-approximation

- Explicitly construct n set cover instances where each instance corresponds to a source vertex s.
- Takes $O(n^3)$ time to construct all instances, and another $O(n^3)$ time to run greedy set cover on them.

Can we $O(\ln n)$-approximate α-SNP any faster?

Faster Algorithms via Set Cover Connection

- For a source s and $u \in P$,
  checking if $t \in Z(s,u) \iff d(u,t) < d(s,t)/\alpha$.
- Set membership queries take $O(1)$ time!
- We exploit this and bypass an explicit construction of set cover instances.

Result 3 [K, Padaki, Waingarten '25]
There is an $O(\ln n)$-approximation algorithm for α-SNP that runs in $\tilde{O}(n^2 \cdot OPT)$ time where OPT is optimal max degree.

Set Cover in Membership Query Model

Lemma : In membership query model, $O(\ln n)$-approximation to Set Cover can be achieved in $\tilde{O}((m+n)\cdot OPT)$ time.

- In our setting, for each source s, we have $m = n - 1$, hence each set cover instance can be solved in $\tilde{O}(n \cdot OPT)$ time.

Plan:

- Greedy set cover algorithm relies on repeatedly choosing a heavy set: a set that covers $\Omega(\frac{n}{OPT})$ elements.
- If we can show this can be done in $\tilde{O}(m)$ time, the result follows as greedy chooses $O(OPT \cdot \ln n)$ sets.

Finding a Heavy Set

Idea: Let us consider a simplified setting where every set in OPT covers $\Theta(n/OPT)$ elements.

- Sample a family $F' \subseteq F$ of $\Theta(\frac{m}{OPT} \cdot \log m)$ sets: F' almost certainly a heavy set!
- Sample a set $U' \subseteq U$ of $\Theta(OPT \cdot \log m)$ elements.
- For each set $S \in F'$,
  - use membership queries to find $|S \cap U'|$.
  - set S is heavy iff $|S \cap U'| = \Theta(\log m)$.
- Since each membership query takes $O(1)$ time, we can identify a heavy set in $O(|F'| \cdot |U'|)$= $\tilde{O}(m)$ time.

The Small OPT Regime

- Previous algorithm works well when OPT is small.
- It solves α-SNP in $\tilde{O}(n^2)$ time when $OPT = \tilde{O}(1)$. This can in fact be shown to be optimal in the small OPT regime.

Result 4 [K, Padaki, Waingarten '25]
Any algorithm that achieves $o(n)$-approximation to 1-SNP must make $\Omega(n^2)$ queries to metric d.

- There is a 1-SNP instance with $OPT = 3$ where outputting any $o(n)$–approximate solution requires $\Omega(n^2)$ queries.

Fast Bicriteria Navigability when OPT is Large

- However, as OPT approaches n, the previous algorithm degenerates to the baseline $O(n^3)$ time algorithm.
- Can we beat the baseline algorithm when OPT is large?

- Yes, if we settle for a bicriteria-approximation!
  - We will design a solution to α-SNP and compare its performance to an optimal solution to 2α-SNP.
  - We will refer to this problem as $(\alpha, 2\alpha)$-SNP.

Result 5 [K, Padaki, Waingarten '25]
There is an $\tilde{O}(n^\omega)$ time algorithm for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNP.

Bicriteria Navigability via Matrix Multiplication

- The set cover approach to α-SNP iteratively builds a graph $G(P, E)$ using two steps repeatedly:
  - Identify pairs s, t that do not yet satisfy α-navigability in $G(P, E)$.
  - Greedily add edges (sets) that take care of many unsatisfied pairs.

- Both tasks are bottleneck when max degree is large.
- In particular, even verifying if the current solution is feasible requires checking:

  $\forall s \neq t \in P$, and $(s, u) \in E$ if $d(u, t) < d(s, t)/\alpha$.
- Takes $O(n^2 \cdot \deg(G))$ time: becomes $O(n^3)$ as $\deg(G) \rightarrow n$.

Can we at least speed up verification?

Batch Verification of α-Navigability

- Suppose $d(s,t) = r$, and we want to verify if it is covered.
- Define matrices $A, B_r \in \{0,1\}^{n \times n}$ as below:
  - $A[s,u] = 1$ iff $(s,u) \in E$. (Adjacency in G)
  - $B_r[u,t] = 1$ iff $d(u,t) < d(s,t)/\alpha$. (Small distances in P)
- Then the pair s, t is covered in G iff

$$(A \cdot B_r)[s,t] \neq 0 .$$

- So verification for all pairs s, t at distance r can be done by a single matrix multiplication.
- We can group distances into geometric ranges and verify $\tilde{O}(\frac{\ln \Delta}{\epsilon})$ distinct distance scales.
- Thus (approximate) verification can be done in $\tilde{O}(n^\omega)$ time.

Batch Verification to α-Navigability

- Let $K_{2\alpha}$ be the optimal max degree in a (2α)-navigable graph on P.

Lemma: If each vertex randomly samples $K_{2\alpha}$ edges to its uncovered neighbors, then w.h.p. total number of pairs violating α–navigability constraint reduces by an $O(1)$ factor.

- So repeating the verification + random sampling $O(\ln n)$ times gives the desired bicriteria result.

From Batch Verification to α-Navigability

Lemma: If each vertex randomly samples $K_{2\alpha}$ edges to its uncovered neighbors, then w.h.p. total number of pairs violating α–navigability constraint reduces by an $O(1)$ factor.

- Fix source s, and an edge (s,u), that covers 2α–navigability constraints from s to points $y_1, y_2, \ldots, y_q$, arranged in increasing order of distance from s.
- Then for any $a \leq q/2$ and $b > q/2$, we have $d(y_a, y_b) \leq d(y_a, u) + d(u, y_b) \leq \frac{d(s,y_a)}{2\alpha} + \frac{d(s,y_b)}{2\alpha} \leq \frac{d(s,y_b)}{\alpha}$
- So every vertex in first half can α–cover every vertex in second half!

Concluding Remarks

- We showed the following:
  - α-SNP is equivalent to set cover and hence $\Theta(\ln n)$-hard to approximate.
  - $O(\ln n)$-approximation for α-SNP in $\tilde{O}(n^2 \cdot OPT)$ time, and $\Omega(n^2)$ time needed even when $OPT = O(1)$.
  - $O(\ln n)$-approximation for $(\alpha, 2\alpha)$-SNP in $\tilde{O}(n^\omega)$ time.
- Parallel work by [Conway et. al. '25] gives stronger version of some of our results:
  - $O(\ln n)$-approximation for 1-SNP in $\tilde{O}(n^2)$ time.
  - $O(\ln n)$-approximation for α-SNP in $\tilde{O}(n^{2.5})$ time.

Thank you !