Sparse Navigable Graphs for Nearest Neighbor Search

Sanjeev Khanna
University of Pennsylvania

Ashwin Padaki
University of Pennsylvania

Erik Waingarten
University of Pennsylvania

(SODA 2026)

Outline of the Talk

1. **Background** on navigability

Outline of the Talk

1. **Background** on navigability
2. **Set Cover View** of navigability

Outline of the Talk

1. **Background** on navigability
2. **Set Cover View** of navigability
3. **Faster Algorithms** for building navigable graphs

Outline of the Talk

1. **Background** on navigability
2. **Set Cover View** of navigability
3. **Faster Algorithms** for building navigable graphs
4. **Lower Bound** on query complexity

Outline of the Talk

1. **Background** on navigability
2. **Set Cover View** of navigability
3. **Faster Algorithms** for building navigable graphs
4. **Lower Bound** on query complexity

Concurrent work with overlapping results!

- Conway, Dhulipala, Farach-Colton, Johnson, Landrum, Musco, Schechter, Suel, Wen. *Efficiently Constructing Sparse Navigable Graphs*. (2025)

Background

(1/4)

Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query $\boldsymbol{q}$, find the point in $\mathbf{P}$ closest to $\boldsymbol{q}$.

Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query $\boldsymbol{q}$, find the point in $\mathbf{P}$ closest to $\boldsymbol{q}$.

Simplest way to solve NNS?

Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query $\boldsymbol{q}$, find the point in $\mathbf{P}$ closest to $\boldsymbol{q}$.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹

Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query $\boldsymbol{q}$, find the point in $\mathbf{P}$ closest to $\boldsymbol{q}$.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺

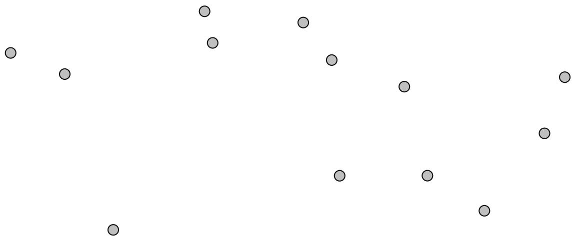
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



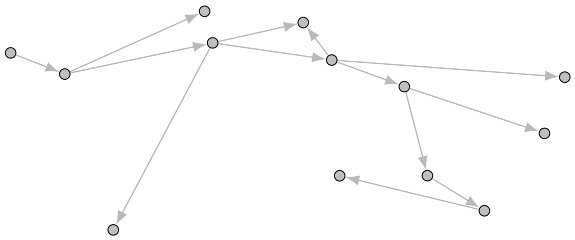
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query $\boldsymbol{q}$, find the point in $\mathbf{P}$ closest to $\boldsymbol{q}$.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



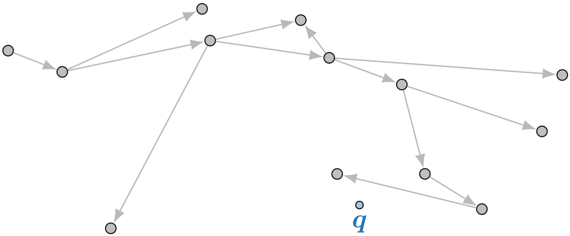
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



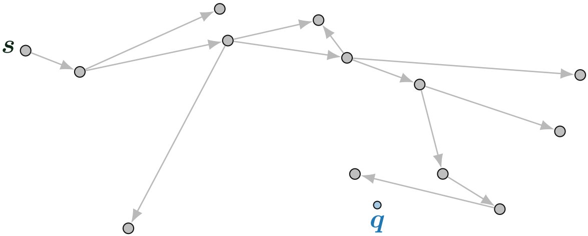
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



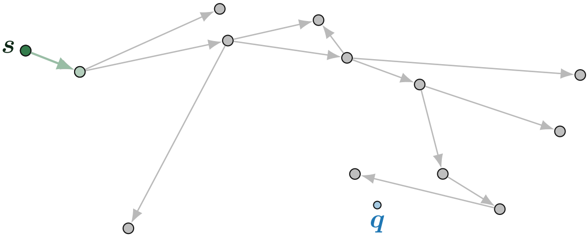
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



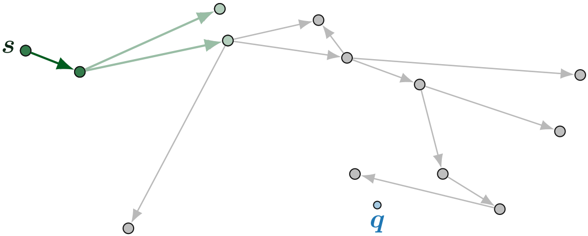
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



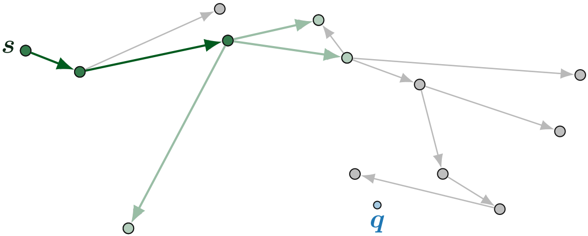
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



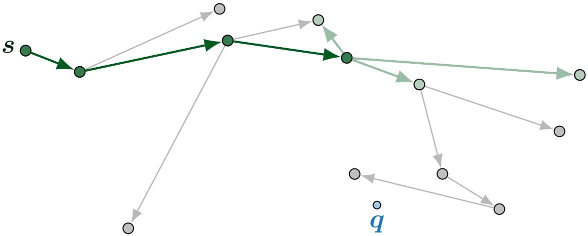
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



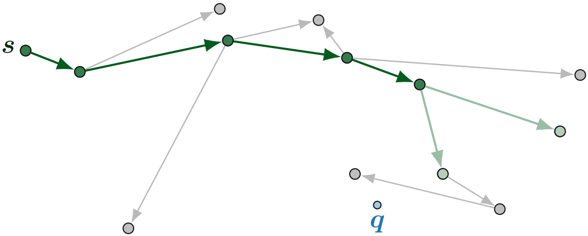
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query $\boldsymbol{q}$, find the point in $\mathbf{P}$ closest to $\boldsymbol{q}$.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



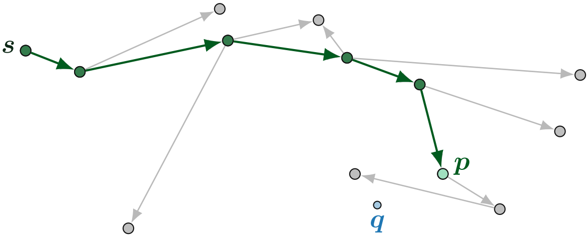
Nearest Neighbor Search

Nearest Neighbor Search (NNS)

Given dataset $\mathbf{P}$ and query q, find the point in $\mathbf{P}$ closest to q.

Simplest way to solve NNS?

- **Brute force:** scan through $\mathbf{P} \longrightarrow O(n)$ time ☹
- **More clever:** build a graph on $\mathbf{P}$ + *greedy search* ☺



Why Graphs? It's a Small World

Real-world networks exhibit *small world* behavior

Why Graphs? It's a Small World

Real-world networks exhibit *small world* behavior

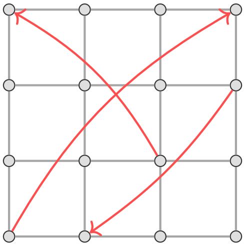
Milgram (1969). Packages from Nebraska $\longrightarrow$ Boston

- "Pass to a friend closer to Boston"
- Median chain length: only six!

Navigability in Small-World Graphs

Kleinberg (2000). Formalization of Milgram's experiment

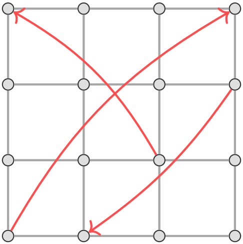
- **Model:** Grid + random long-range edges
- **Rule:** Move to neighbor closest to the target



Navigability in Small-World Graphs

Kleinberg (2000). Formalization of Milgram's experiment

- **Model:** Grid + random long-range edges
- **Rule:** Move to neighbor closest to the target

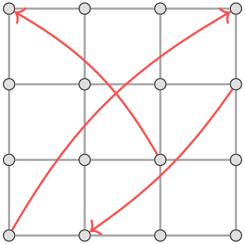


Q. Can we make *arbitrary data* navigable for NNS?

Navigability in Small-World Graphs

Kleinberg (2000). Formalization of Milgram's experiment

- **Model:** Grid + random long-range edges
- **Rule:** Move to neighbor closest to the target



Q. Can we make *arbitrary data* navigable for NNS?

(Motivation behind modern NNS heuristics!)

Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

Navigability

$\mathbf{G}$ is *navigable* if for all $\boldsymbol{s} \neq \boldsymbol{t} \in \mathbf{P}$, $\exists(\boldsymbol{s}, u) \in E$ such that

$$\mathsf{d}(u, \boldsymbol{t}) < \mathsf{d}(\boldsymbol{s}, \boldsymbol{t}).$$

Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

Navigability

$\mathbf{G}$ is *navigable* if for all $\boldsymbol{s} \neq \boldsymbol{t} \in \mathbf{P}$, $\exists(\boldsymbol{s}, u) \in E$ such that

$$\mathsf{d}(u, \boldsymbol{t}) < \mathsf{d}(\boldsymbol{s}, \boldsymbol{t}).$$

"Greedy search eventually reaches target"

Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

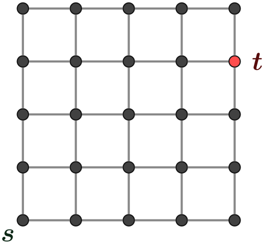
Navigability

$\mathbf{G}$ is *navigable* if for all $\boldsymbol{s} \neq \boldsymbol{t} \in \mathbf{P}$, $\exists(\boldsymbol{s}, u) \in E$ such that

$$\mathsf{d}(u, \boldsymbol{t}) < \mathsf{d}(\boldsymbol{s}, \boldsymbol{t}).$$

“Greedy search eventually reaches target”

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

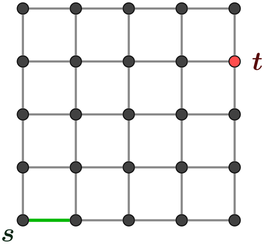
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists (s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

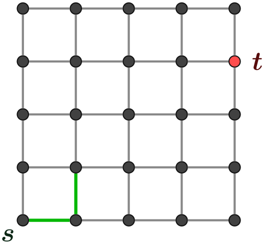
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists (s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

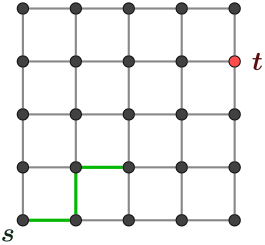
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists (s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

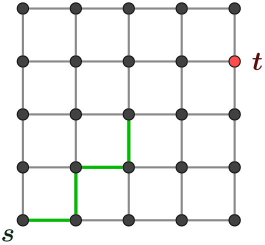
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists (s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

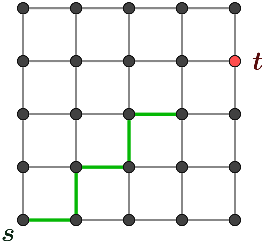
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists (s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

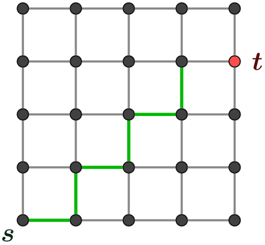
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists(s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Navigability in General Metrics

Dataset $\mathbf{P}$ with metric d; directed graph $\mathbf{G} = (\mathbf{P}, E)$.

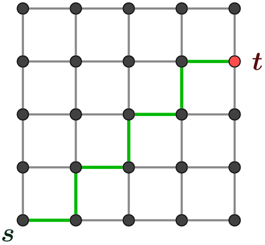
Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t \in \mathbf{P}$, $\exists (s, u) \in E$ such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

"Greedy search eventually reaches target"

Example: Grid graph in $\mathbb{Z}^2$



Nearest Neighbor Search?

Issue. Navigability $\not\Longrightarrow$ nearest neighbor search

Nearest Neighbor Search?

Issue. Navigability $\not\Rightarrow$ nearest neighbor search

Approximate Nearest Neighbor (ANN)

Return *any* c-approximate nearest neighbor $\boldsymbol{p}$ of query $\boldsymbol{q}$:

$$\mathsf{d}(\boldsymbol{p}, \boldsymbol{q}) \leq c \cdot \mathsf{d}(\boldsymbol{p}^*, \boldsymbol{q}).$$

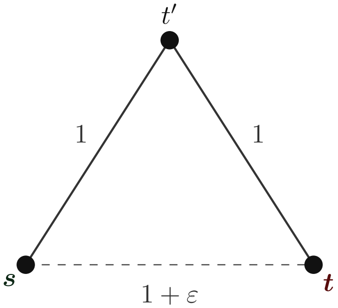
Nearest Neighbor Search?

Issue. Navigability $\not\Rightarrow$ nearest neighbor search

Approximate Nearest Neighbor (ANN)

Return *any* c-approximate nearest neighbor $\boldsymbol{p}$ of query $\boldsymbol{q}$:

$$\mathsf{d}(\boldsymbol{p}, \boldsymbol{q}) \leq c \cdot \mathsf{d}(\boldsymbol{p}^*, \boldsymbol{q}).$$



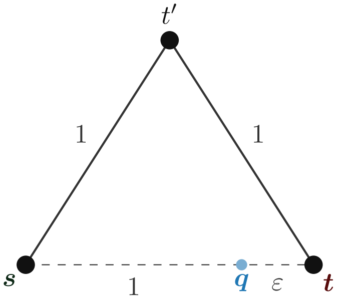
Nearest Neighbor Search?

Issue. Navigability $\not\Rightarrow$ nearest neighbor search

Approximate Nearest Neighbor (ANN)

Return *any* c-approximate nearest neighbor $\boldsymbol{p}$ of query $\boldsymbol{q}$:

$$\mathsf{d}(\boldsymbol{p}, \boldsymbol{q}) \leq c \cdot \mathsf{d}(\boldsymbol{p}^*, \boldsymbol{q}).$$



Nearest Neighbor Search?

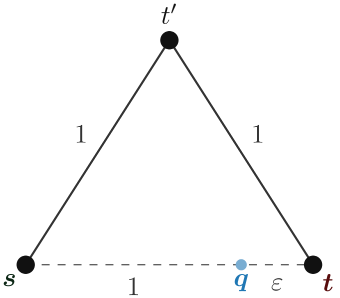
Issue. Navigability $\not\Longrightarrow$ nearest neighbor search

Approximate Nearest Neighbor (ANN)

Return *any* c-approximate nearest neighbor $\boldsymbol{p}$ of query $\boldsymbol{q}$:

$$\mathsf{d}(\boldsymbol{p}, \boldsymbol{q}) \leq c \cdot \mathsf{d}(\boldsymbol{p}^*, \boldsymbol{q}).$$

$\mathrm{GreedySearch}(\boldsymbol{s}, \boldsymbol{q}) \to (1/\varepsilon)$-ANN



α-Navigability: a small (world) fix!

Navigability

$\mathbf{G}$ is *navigable* if for all $s \neq t$, $\exists$ edge (s, u) such that

$$\mathsf{d}(u, t) < \mathsf{d}(s, t).$$

α-Navigability: a small (world) fix!

α-Navigability [Indyk-Xu, 2023]

$\mathbf{G}$ is *α-navigable* if for all $\boldsymbol{s} \neq \boldsymbol{t} \in \mathbf{P}$, $\exists$ edge $(\boldsymbol{s}, u)$ s.t.

$$\mathsf{d}(u, \boldsymbol{t}) < \mathsf{d}(\boldsymbol{s}, \boldsymbol{t})/\alpha.$$

α-Navigability: a small (world) fix!

α-Navigability [Indyk-Xu, 2023]

$\mathbf{G}$ is *α-navigable* if for all $\boldsymbol{s} \neq \boldsymbol{t} \in \mathbf{P}$, $\exists$ edge $(\boldsymbol{s}, u)$ s.t.

$$\mathsf{d}(u, \boldsymbol{t}) < \mathsf{d}(\boldsymbol{s}, \boldsymbol{t})/\alpha.$$

Surprising Theorem! [Indyk–Xu, 2023]

$\forall \alpha > 1$, if $\mathbf{G}$ is α-navigable, then $\mathrm{GreedySearch}$ returns a

$$\left(\frac{\alpha+1}{\alpha-1} + \varepsilon\right)\text{–ANN in } O(\log(\Delta/\varepsilon)) \text{ hops.}$$

Takeaway: sparse α-navigable graphs $\implies$ fast ANN!

Building α-Navigable Graphs

Heuristics: DiskANN, HNSW, etc.

Building α-Navigable Graphs

Heuristics: DiskANN, HNSW, etc.

Theory:

- Slow-DiskANN $\to$ α-nav graph with $\deg_{\mathsf{max}} \leq (4\alpha)^{\lambda(P)}$ [IX23]

Building α-Navigable Graphs

Heuristics: DiskANN, HNSW, etc.

Theory:

- Slow-DiskANN $\rightarrow$ α-nav graph with $\deg_{\text{max}} \leq (4\alpha)^{\lambda(P)}$ [IX23]
- $\exists$ 1-navigable graph of size $\leq \tilde{O}(n^{1.5})$ [DGM$^+$24]

Building α-Navigable Graphs

Heuristics: DiskANN, HNSW, etc.

Theory:

- Slow-DiskANN $\rightarrow$ α-nav graph with $\deg_{\text{max}} \leq (4\alpha)^{\lambda(P)}$ [IX23]
- $\exists$ 1-navigable graph of size $\leq \tilde{O}(n^{1.5})$ [DGM+24]

Our Paper:

Sparsest Navigable Graph (SNG)

Given dataset $(\mathbf{P}, \mathrm{d})$ and $\alpha \geq 1$,

- What is the sparsest α-navigable graph on $\mathbf{P}$?
- How fast can we compute (or approximate) it?

Building α-Navigable Graphs

Heuristics: DiskANN, HNSW, etc.

Theory:

- Slow-DiskANN $\rightarrow$ α-nav graph with $\deg_{\max} \leq (4\alpha)^{\lambda(P)}$ [IX23]
- $\exists$ 1-navigable graph of size $\leq \tilde{O}(n^{1.5})$ [DGM+24]

Our Paper:

Sparsest Navigable Graph (SNG)

Given dataset $(\mathbf{P}, \mathsf{d})$ and $\alpha \geq 1$,

- What is the sparsest α-navigable graph on $\mathbf{P}$?
- How fast can we compute (or approximate) it?

Sparsity objective: minimize maximum degree

Negative Result for Slow-DiskANN

Slow-DiskANN $\rightarrow$ α-nav graph with $\deg_{\mathsf{max}} \leq (4\alpha)^{\lambda(P)}$ [IX23]

($\lambda(P) :=$ doubling dimension of P)

Negative Result for Slow-DiskANN

Slow-DiskANN $\rightarrow \alpha$-nav graph with $\deg_{\text{max}} \leq (4\alpha)^{\lambda(P)}$ [IX23]

($\lambda(P) :=$ doubling dimension of P)

Result 1

There is a dataset $\mathbf{P}$ where:

- $\exists\ 1$-navigable graph of max-degree $O(\log n)$
- Slow-DiskANN outputs graph of degree $\Theta(n)$

$\Longrightarrow$ Slow-DiskANN gives $\tilde{\Omega}(n)$-approximation (very bad!)

Set Cover View

(2/4)

SNG $\longrightarrow$ Set Cover

α-Navigability from source s

For all $t \neq s$, there exists an edge (s, u) with

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

SNG $\longrightarrow$ Set Cover

α-Navigability from source s

For all $t \neq s$, there exists an edge (s, u) with

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

Equivalent Set Cover instance.

Elements	$\mathbf{P} \setminus \{s\}$
Sets	$Z(s, u) := \{t \mid \mathsf{d}(u, t) < \mathsf{d}(s, t)/\alpha\}$
Cover size	$\deg(s)$

SNG $\longrightarrow$ Set Cover

α-Navigability from source s

For all $t \neq s$, there exists an edge (s, u) with

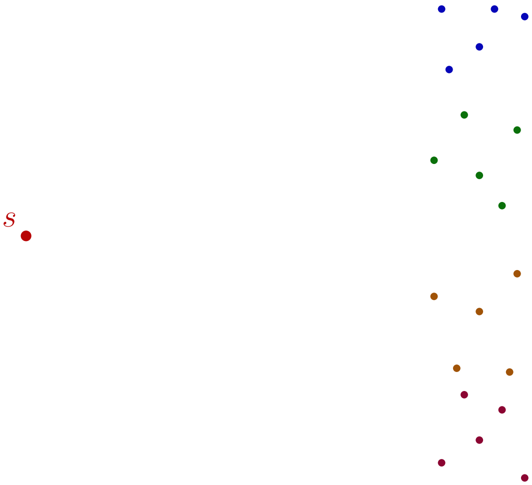
$$\mathsf{d}(u, t) < \mathsf{d}(s, t)/\alpha.$$

Equivalent Set Cover instance.

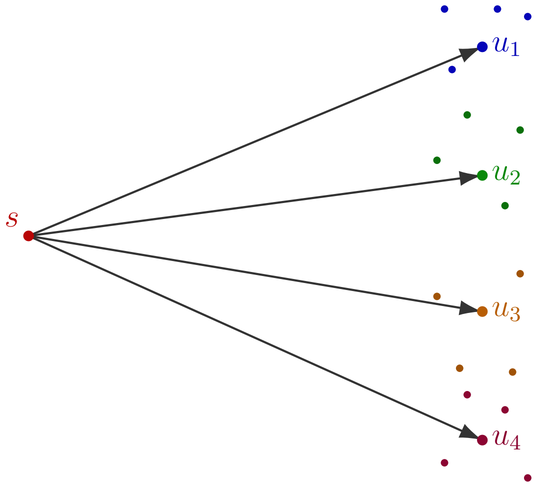
Elements	$\mathbf{P} \setminus \{s\}$
Sets	$Z(s, u) := \{t \mid \mathsf{d}(u, t) < \mathsf{d}(s, t)/\alpha\}$
Cover size	$\deg(s)$

Key: α-navigability $\equiv$ n Set Cover instances

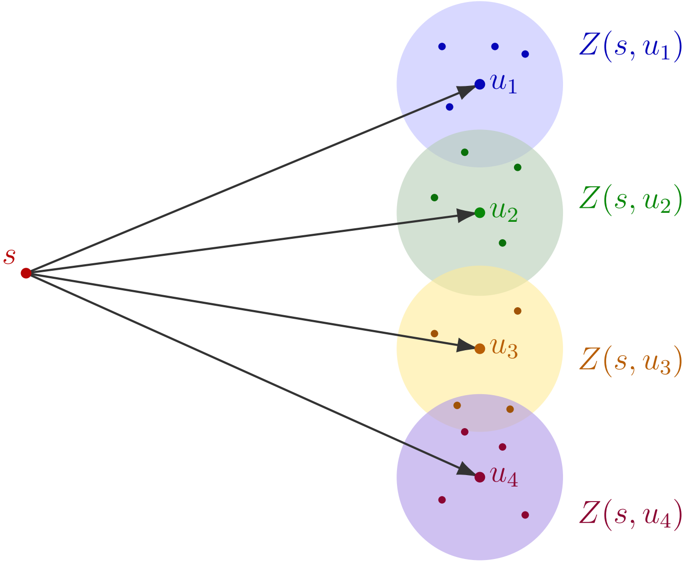
SNG $\longrightarrow$ Set Cover



SNG $\longrightarrow$ Set Cover



SNG $\longrightarrow$ Set Cover



Algorithm via Greedy Set Cover

Key: α-navigability $\equiv$ n Set Cover instances

Algorithm via Greedy Set Cover

Key: α-navigability $\equiv$ n Set Cover instances

Each instance:

- $n-1$ elements, $n-1$ sets

Algorithm via Greedy Set Cover

Key: α-navigability $\equiv$ n Set Cover instances

Each instance:

- $n-1$ elements, $n-1$ sets
- Construction: $O(n^2)$ time

Algorithm via Greedy Set Cover

Key: α-navigability $\equiv$ n Set Cover instances

Each instance:

- $n-1$ elements, $n-1$ sets
- Construction: $O(n^2)$ time
- Greedy Alg: $O(n^2)$ time for $(\ln n + 1)$-approximation

Algorithm via Greedy Set Cover

Key: α-navigability $\equiv$ n Set Cover instances

Each instance:

- $n-1$ elements, $n-1$ sets
- Construction: $O(n^2)$ time
- Greedy Alg: $O(n^2)$ time for $(\ln n + 1)$-approximation

Result 2A

$O(n^3)$-time algorithm for $(\ln n + 1)$-approximation to SNG.

Hardness via Set Cover

Result 2B

NP-hard to compute $(c \ln n)$-approximation to SNG, for some $c > 0$.

Idea. Encode Set Cover as a navigability condition!

Baseline Results via Set Cover

Result 2A

$O(n^3)$-time algorithm for $(\ln n + 1)$-approximation to SNG.

Result 2B

NP-hard to compute $(c \ln n)$-approximation to SNG, for some $c > 0$.

Baseline Results via Set Cover

Result 2A

$O(n^3)$-time algorithm for $(\ln n + 1)$-approximation to SNG.

Result 2B

NP-hard to compute $(c \ln n)$-approximation to SNG, for some $c > 0$.

Q. Can we compute $O(\ln n)$-approximation in time $o(n^3)$?

Faster Algorithms

(3/4)

Membership Set Cover

Recall. Building all sets $Z(s, u) \rightarrow O(n^3)$ time

Membership Set Cover

Recall. Building all sets $Z(s, u) \to O(n^3)$ time

But, checking $t \in Z(s, u) \;\equiv\;$ two calls to $\mathsf{d}(\cdot, \cdot)$

$$\mathsf{d}(u, t) < \mathsf{d}(s, t)/\alpha$$

Membership Set Cover

Recall. Building all sets $Z(s,u) \to O(n^3)$ time

But, checking $t \in Z(s,u) \quad \equiv \quad$ two calls to $\mathsf{d}(\cdot,\cdot)$

$$\mathsf{d}(u,t) < \mathsf{d}(s,t)/\alpha$$

Set Cover (Membership Model)

Access input only via queries “is element x in set S?”

Algorithm via Membership Set Cover

n elements, m sets, $k =$ min cover size

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Algorithm via Membership Set Cover

n elements, m sets, $k =$ min cover size

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

$\tilde{O}(mk + nk) \longrightarrow \tilde{O}(n \cdot \deg(s))$

Algorithm via Membership Set Cover

n elements, m sets, $k =$ min cover size

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

$$\tilde{O}(mk + nk) \longrightarrow \tilde{O}(n \cdot \deg(s))$$

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

($\mathrm{OPT}_{\text{size}} :=$ min size of any α-navigable graph)

Consequences of Result 3

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

($\mathrm{OPT}_{\text{size}}$:= min size of any α-navigable graph)

Consequences of Result 3

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

($\mathrm{OPT}_{\text{size}} :=$ min size of any α-navigable graph)

- $\mathrm{OPT}_{\text{size}} = \tilde{O}(n) \implies \tilde{O}(n^2)$ runtime

Consequences of Result 3

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\mathsf{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

$(\mathrm{OPT}_{\mathsf{size}}$:= min size of any α-navigable graph)

- $\mathrm{OPT}_{\mathsf{size}} = \tilde{O}(n) \implies \tilde{O}(n^2)$ runtime
- $\alpha = 1 \implies \mathrm{OPT}_{\mathsf{size}} = \tilde{O}(n^{1.5})$ [DGM+24]

  $\implies \tilde{O}(n^{2.5})$ runtime

Fast Algorithm for Membership Set Cover

n elements, m sets, $k =$ min cover size

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Fast Algorithm for Membership Set Cover

n elements, m sets, $k =$ min cover size

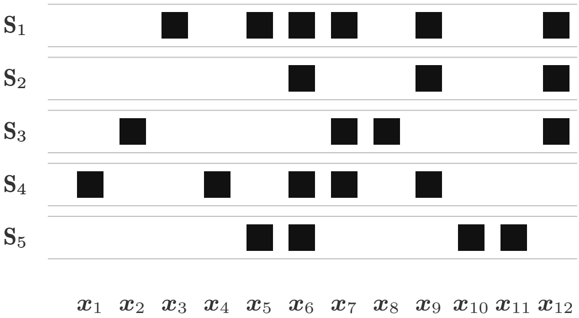
Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Idea. Simulate greedy via random sampling

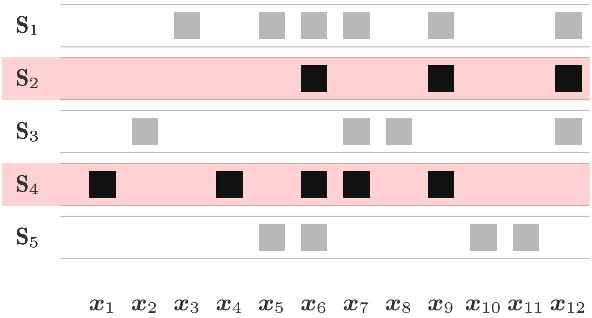
- Greedy: heaviest set covers $\geq (1/k)$-fraction of elements

FindHeavySet: Illustration



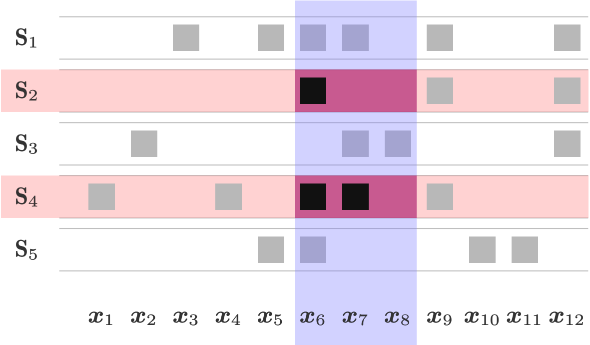
- Sets $\mathbf{S}_1, \ldots, \mathbf{S}_5$, elements $x_1, \ldots, x_{12}$

FindHeavySet: Illustration



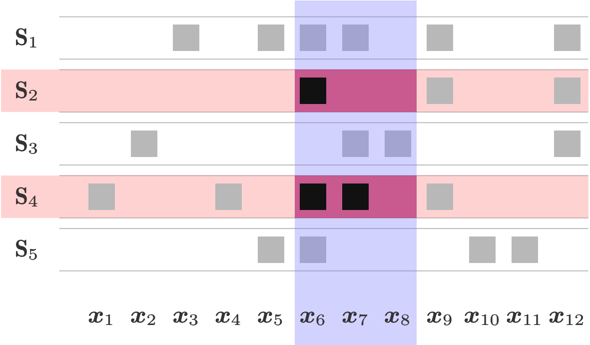
- Sets $\mathbf{S}_1, \ldots, \mathbf{S}_5$, elements $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{12}$
- Set sample: $\{\mathbf{S}_2, \mathbf{S}_4\}$

FindHeavySet: Illustration



- Sets $\mathbf{S}_1, \ldots, \mathbf{S}_5$, elements $x_1, \ldots, x_{12}$
- Set sample: $\{\mathbf{S}_2, \mathbf{S}_4\}$
- Element sample: $\{x_6, x_7, x_8\}$

FindHeavySet: Illustration



- Sets $S_1, \ldots, S_5$, elements $x_1, \ldots, x_{12}$
- Set sample: $\{S_2, S_4\}$
- Element sample: $\{x_6, x_7, x_8\}$
- S_4 hits many elements $\Longrightarrow$ good set to pick!

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

Imagine: $\exists\, k$ heavy sets, weight $\Omega(1/k)$

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

Imagine: $\exists\, k$ heavy sets, weight $\Omega(1/k)$

- Sample $\tilde{O}(m/k)$ sets $\longrightarrow$ one heavy w.h.p.

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

Imagine: $\exists\, k$ heavy sets, weight $\Omega(1/k)$

- Sample $\tilde{O}(m/k)$ sets $\longrightarrow$ one heavy w.h.p.
- Sample $\Theta(k \ln n)$ uncovered elements

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

Imagine: $\exists\, k$ heavy sets, weight $\Omega(1/k)$

- Sample $\tilde{O}(m/k)$ sets $\longrightarrow$ one heavy w.h.p.
- Sample $\Theta(k \ln n)$ uncovered elements
- Set hits $\Omega(\ln n)$ elements $\iff$ **heavy**

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

Imagine: $\exists\, k$ heavy sets, weight $\Omega(1/k)$

- Sample $\tilde{O}(m/k)$ sets $\longrightarrow$ one heavy w.h.p.
- Sample $\Theta(k \ln n)$ uncovered elements
- Set hits $\Omega(\ln n)$ elements $\iff$ **heavy**

$\tilde{O}(m)$ queries to find heavy set

FindHeavySet: Procedure

Goal. Find a set covering $\Omega(1/k)$-fraction of uncovered elements

Imagine: $\exists\, k$ heavy sets, weight $\Omega(1/k)$

- Sample $\tilde{O}(m/k)$ sets $\longrightarrow$ one heavy w.h.p.
- Sample $\Theta(k \ln n)$ uncovered elements
- Set hits $\Omega(\ln n)$ elements $\iff$ **heavy**

$\tilde{O}(m)$ queries to find heavy set

vs.

$\Omega(mn)$ to find heaviest

Final Algorithm

Call FindHeavySet until full cover $\longrightarrow O(k \ln n)$ rounds

Final Algorithm

Call FindHeavySet until full cover $\longrightarrow O(k \ln n)$ rounds

Queries per round:

- $\tilde{O}(m)$ to find heavy set

Final Algorithm

Call FindHeavySet until full cover $\longrightarrow O(k \ln n)$ rounds

Queries per round:

- $\tilde{O}(m)$ to find heavy set
- $O(n)$ to update uncovered elements

Final Algorithm

Call FindHeavySet until full cover $\longrightarrow O(k \ln n)$ rounds

Queries per round:

- $\tilde{O}(m)$ to find heavy set
- $O(n)$ to update uncovered elements

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Final Algorithm

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Final Algorithm

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

Final Algorithm

Lemma

$O(\ln n)$-approx Set Cover (membership) in $\tilde{O}(mk + nk)$ time.

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

Q. What if $\mathrm{OPT}_{\text{size}} = \tilde{\Omega}(n^2)$?

Bicriteria Approximation to SNG

Bicriteria Approximation to SNG

Given $\alpha \le \beta$, build α-navigable graph $\mathbf{G}$.

- Let $k_\beta :=$ max-degree of sparsest β-navigable graph
- **Guarantee.** $\deg(\mathbf{G}) \le$ (approx factor) $\times\ k_\beta$

Bicriteria Approximation to SNG

Bicriteria Approximation to SNG

Given $\alpha \leq \beta$, build α-navigable graph $\mathbf{G}$.

- Let $k_\beta :=$ max-degree of sparsest β-navigable graph
- **Guarantee.** $\deg(\mathbf{G}) \leq$ (approx factor) $\times k_\beta$

Result 4 (informal)

$\tilde{O}(n^\omega)$-time algorithm for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG.

($\omega \approx 2.37 =$ matrix multiplication exponent)

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Naive:

- For all $s \neq t \in \mathbf{P}$ and $(s, u) \in E$, check

$$\mathsf{d}(u, t) < \mathsf{d}(s, t)/\alpha$$

- $\Theta(n^2 \cdot \deg(\mathbf{G})) \longrightarrow$ potentially $\Omega(n^3)$ time!

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Better: batch verification!

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Better: batch verification!

- Let $r = \mathsf{d}(s, t)$ and $A, B_r \in \{0,1\}^{n \times n}$

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Better: batch verification!

- Let $r = \mathsf{d}(s,t)$ and $A, B_r \in \{0,1\}^{n\times n}$

$$A[s,u] = 1 \iff (s,u) \in E \qquad \text{(adjacency in } \mathbf{G}\text{)}$$

$$B_r[u,t] = 1 \iff \mathsf{d}(u,t) < r/\alpha. \qquad \text{(small dists in } \mathbf{P}\text{)}$$

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Better: batch verification!

- Let $r = \mathsf{d}(s,t)$ and $A, B_r \in \{0,1\}^{n \times n}$

$$A[s,u] = 1 \iff (s,u) \in E \qquad \text{(adjacency in } \mathbf{G}\text{)}$$

$$B_r[u,t] = 1 \iff \mathsf{d}(u,t) < r/\alpha. \qquad \text{(small dists in } \mathbf{P}\text{)}$$

- (s,t) constraint satisfied by $\mathbf{G} \iff (A \cdot B_r)[s,t] \neq 0$

Verification via Matrix Multiplication

How to even verify that $\mathbf{G}$ is α-navigable?

Better: batch verification!

- Let $r = \mathsf{d}(s,t)$ and $A, B_r \in \{0,1\}^{n\times n}$

$$A[s,u] = 1 \iff (s,u) \in E \qquad \text{(adjacency in } \mathbf{G}\text{)}$$

$$B_r[u,t] = 1 \iff \mathsf{d}(u,t) < r/\alpha. \qquad \text{(small dists in } \mathbf{P}\text{)}$$

- (s,t) constraint satisfied by $\mathbf{G} \iff (A \cdot B_r)[s,t] \neq 0$

Lemma

If $\mathbf{P}$ has L distances, we can verify α-navigability in time $O(L \cdot n^{\omega})$.

Bicriteria Algorithm Sketch

Lemma

If $\mathbf{P}$ has L distances, we can verify α-navigability in time $O(L \cdot n^{\omega})$.

Algorithm: Discretize distances in $\mathbf{P}$, then repeat:

Bicriteria Algorithm Sketch

Lemma

If $\mathbf{P}$ has L distances, we can verify α-navigability in time $O(L \cdot n^{\omega})$.

Algorithm: Discretize distances in $\mathbf{P}$, then repeat:

1. **Sample** $k_{2\alpha}$ uncovered constraints $\longrightarrow$ add edges

Bicriteria Algorithm Sketch

Lemma

If $\mathbf{P}$ has L distances, we can verify α-navigability in time $O(L \cdot n^{\omega})$.

Algorithm: Discretize distances in $\mathbf{P}$, then repeat:

1. **Sample** $k_{2\alpha}$ uncovered constraints $\longrightarrow$ add edges
2. **Verify** $\longrightarrow$ remaining uncovered constraints

Bicriteria Algorithm Sketch

Lemma

If $\mathbf{P}$ has L distances, we can verify α-navigability in time $O(L \cdot n^\omega)$.

Algorithm: Discretize distances in $\mathbf{P}$, then repeat:

1. **Sample** $k_{2\alpha}$ uncovered constraints $\longrightarrow$ add edges
2. **Verify** $\longrightarrow$ remaining uncovered constraints

Key Lemma: Each round eliminates constant fraction of constraints

Bicriteria Algorithm Sketch

Lemma

If $\mathbf{P}$ has L distances, we can verify α-navigability in time $O(L \cdot n^{\omega})$.

Algorithm: Discretize distances in $\mathbf{P}$, then repeat:

1. **Sample** $k_{2\alpha}$ uncovered constraints $\longrightarrow$ add edges
2. **Verify** $\longrightarrow$ remaining uncovered constraints

Key Lemma: Each round eliminates constant fraction of constraints

Result 4 (informal)

$\tilde{O}(n^{\omega})$-time algorithm for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG.

Faster Algorithms for SNG

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\mathsf{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

Result 4 (informal)

$\tilde{O}(n^{\omega})$-time algorithm for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG.

Faster Algorithms for SNG

Result 3

$\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time algorithm for $O(\ln n)$-approximation to SNG.

Result 4 (informal)

$\tilde{O}(n^{\omega})$-time algorithm for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG.

Q. Even faster algorithms for worse approximation?

Lower Bound

(4/4)

Simple $\Omega(n^2)$ Lower Bound

Result 5

$\Omega(n^2)$ queries to $\mathsf{d}(\cdot,\cdot)$ needed for *any* $o(n)$-approximation to SNG.

Simple $\Omega(n^2)$ Lower Bound

Result 5

$\Omega(n^2)$ queries to $\mathsf{d}(\cdot, \cdot)$ needed for *any* $o(n)$-approximation to SNG.

Idea. Navigability $\implies$ graph contains minimum-distance edge

Simple $\Omega(n^2)$ Lower Bound

Result 5

$\Omega(n^2)$ queries to $\mathsf{d}(\cdot, \cdot)$ needed for *any* $o(n)$-approximation to SNG.

Idea. Navigability $\implies$ graph contains minimum-distance edge

- Fix a metric with constant-degree navigable graph
- Shrink a random distance $\longrightarrow$ hidden shortcut

Perturbed Path Metric

Metric on $[n]$:

$$\mathsf{d}(i,j) = 1 + \frac{|i-j|}{n-1}$$

Perturbation: sample $(\boldsymbol{i}^*, \boldsymbol{j}^*)$ at random, then update

$$\mathsf{d}(\boldsymbol{i}^*, \boldsymbol{j}^*) \longleftarrow 1$$

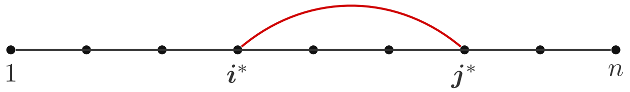
Perturbed Path Metric

$$\mathsf{d}(i,j) = 1 + \frac{|i-j|}{n-1} \in [1,2], \qquad \mathsf{d}(\boldsymbol{i}^*, \boldsymbol{j}^*) = 1$$

Perturbed Path Metric

$$\mathsf{d}(i,j) = 1 + \frac{|i-j|}{n-1} \in [1,2], \qquad \mathsf{d}(\boldsymbol{i}^*, \boldsymbol{j}^*) = 1$$

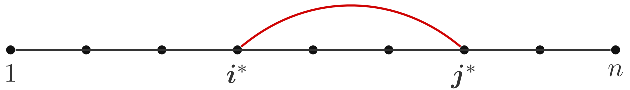
$\exists$ 1-navigable graph of degree 3:



Perturbed Path Metric

$$\mathsf{d}(i,j) = 1 + \frac{|i-j|}{n-1} \in [1,2], \qquad \mathsf{d}(\boldsymbol{i}^*, \boldsymbol{j}^*) = 1$$

$\exists$ 1-navigable graph of degree 3:



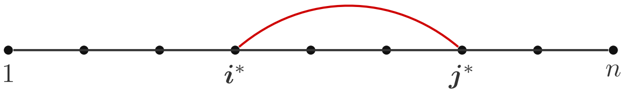
Any 1-navigable graph must contain edge $(\boldsymbol{i}^*, \boldsymbol{j}^*)$

$\implies \Omega(n^2)$ queries for degree $o(n)$

Perturbed Path Metric

$$\mathsf{d}(i,j) = 1 + \frac{|i-j|}{n-1} \in [1,2], \qquad \mathsf{d}(\boldsymbol{i}^*, \boldsymbol{j}^*) = 1$$

$\exists$ 1-navigable graph of degree 3:



Any 1-navigable graph must contain edge $(\boldsymbol{i}^*, \boldsymbol{j}^*)$

$\implies$ $\Omega(n^2)$ queries for degree $o(n)$

[Conway et al. 2025] $\tilde{\Omega}(n^2)$ lower bound in Euclidean via Closest Pair

Navigability Landscape

Results 3, 4, 5

- $\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time for $O(\ln n)$-approximation to SNG
- $\tilde{O}(n^{\omega})$-time for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG
- $\Omega(n^2)$ queries for any $o(n)$-approximation to SNG

Navigability Landscape

Results 3, 4, 5

- $\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time for $O(\ln n)$-approximation to SNG
- $\tilde{O}(n^{\omega})$-time for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG
- $\Omega(n^2)$ queries for any $o(n)$-approximation to SNG

Theorem [Conway et al. 2025]

- $\tilde{O}(n^2)$-time for $O(\ln n)$-approximation for $\alpha = 1$
- $\tilde{O}(n^{2.5})$-time for $O(\ln n)$-approximation for $\alpha > 1$

Navigability Landscape

Results 3, 4, 5

- $\tilde{O}(n \cdot \mathrm{OPT}_{\text{size}})$-time for $O(\ln n)$-approximation to SNG
- $\tilde{O}(n^{\omega})$-time for $O(\ln n)$-approximation to $(\alpha, 2\alpha)$-SNG
- $\Omega(n^2)$ queries for any $o(n)$-approximation to SNG

Theorem [Conway et al. 2025]

- $\tilde{O}(n^2)$-time for $O(\ln n)$-approximation for $\alpha = 1$
- $\tilde{O}(n^{2.5})$-time for $O(\ln n)$-approximation for $\alpha > 1$

Q. Is $\tilde{O}(n^2)$ time possible for $O(\ln n)$-approximation when $\alpha > 1$?

Thanks for Listening!