Near-Linear Query Complexity for Graph Inference

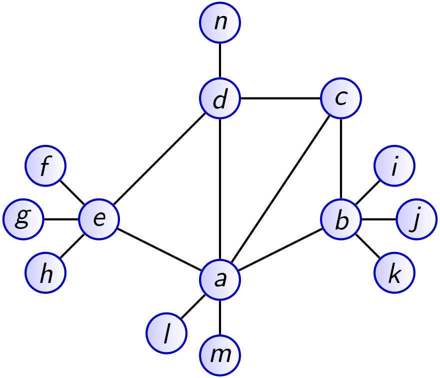
Sampath Kannan[1], Claire Mathieu[2,3], and Hang Zhou[3]

[1]University of Pennsylvania, United States

[2]CNRS, France

[3]École Normale Supérieure de Paris, France

Network tomography



Traceroute

(n, k): $n - d - c - b - k$

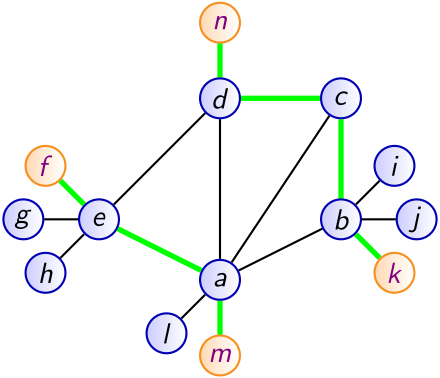
(f, m): $f - e - a - m$

Traceroute blocked by routers

(n, k): $n - \star - \star - \star - k$

(f, m): $f - \star - \star - m$

Network tomography



Traceroute

(n, k): $n - d - c - b - k$

(f, m): $f - e - a - m$

Traceroute blocked by routers

(n, k): $n - \star - \star - \star - k$

(f, m): $f - \star - \star - m$

Graph reconstruction and verification

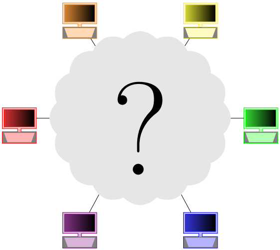
Two QUERY models:

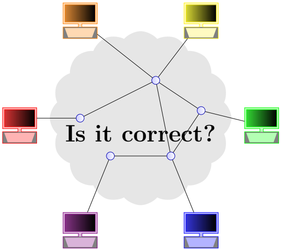
shortest path query: $(u, v) \in V \times V \mapsto$ a shortest u-to-v path

distance query: $(u, v) \in V \times V \mapsto$ length of a shortest u-to-v path

Connected and unweighted graph $G = (V, E)$, where V is known

- E is unknown/guessed and must be reconstructed/verified.
- Minimize the number of queries. Computation is free.





Other models

Network discovery and verification:

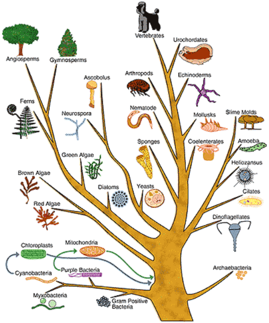
[Beerliova, Eberhard, Erlebach, Hall, Hoffmann, Mihal'ak, Ram, 2006]

query: $u \in V \mapsto \left\{ \text{(length of) a shortest } u\text{-to-}v \text{ path} \right\}_{v \in V}$

Evolutionary tree:

[Hein 1989; King, Zhang, Zhou, 2003; Reyzin, Srivastava, 2007; etc.]

query: leaves $u, v \mapsto$ length of a shortest u-to-v path



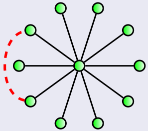
Warm-up: general graphs

$O(n^2)$ algorithm: exhaustive queries

- Query every (u, v)
- Output $\{(u, v) : d(u, v) = 1\}$

$\Omega(n^2)$ lower bound [Reyzin Srivastava 2007]

- $V = \{1, 2, \ldots, n\}$
- $E = \{(1, i) : 2 \leq i \leq n\}$, plus possibly one additional edge (i, j)



Graphs of bounded degree

Reconstruction using distance queries [MZ 2013]

- $\tilde{O}(n\sqrt{n})$ algorithm using Voronoi cell decomposition

Verification

- $n^{1+o(1)}$ greedy algorithm

Reconstruction using shortest path queries

- $n^{1+o(1)}$ greedy algorithm

Open question

Is there a near-linear algorithm for reconstruction using distance queries?

Graphs of bounded degree – side results

$\tilde{O}(n)$ algorithms:

- reconstructing outerplanar graphs using distance queries [MZ 2013]
- reconstructing chordal graphs using distance queries
- verifying bounded treewidth graphs
- reconstructing bounded treewidth graphs using shortest path queries

$\Omega(n \log n / \log \log n)$ lower bound for reconstruction [Gavoille Zwick]

Graphs of bounded degree

Reconstruction using distance queries [MZ 2013]

- $\tilde{O}(n\sqrt{n})$ algorithm using Voronoi cell decomposition

Verification

- $n^{1+o(1)}$ greedy algorithm

Reconstruction using shortest path queries

- $n^{1+o(1)}$ greedy algorithm

Open question

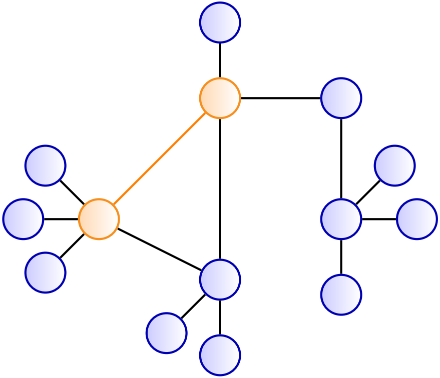
Is there a near-linear algorithm for reconstruction using distance queries?

Verification using distance queries

Input: $\hat{G} = (V, \hat{E})$
Output: whether $\hat{G}$ is correct

Verify edges: Query each pair in $\hat{E}$.

bounded degree $\implies O(n)$ queries



Question: How do we verify $(a, b) \notin \hat{E}$ is a non-edge?

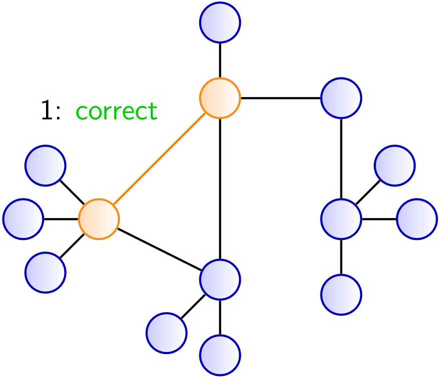
Verification using distance queries

Input: $\hat{G} = (V, \hat{E})$

Output: whether $\hat{G}$ is correct

Verify edges: Query each pair in $\hat{E}$.

bounded degree $\implies$ $O(n)$ queries



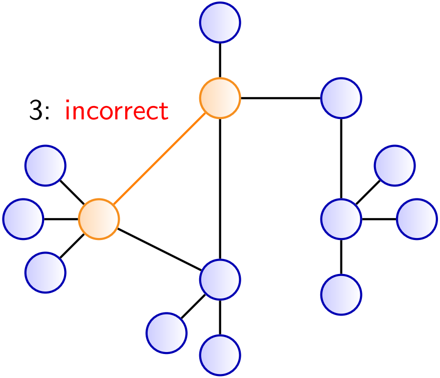
Question: How do we verify $(a, b) \notin \hat{E}$ is a non-edge?

Verification using distance queries

Input: $\hat{G} = (V, \hat{E})$
Output: whether $\hat{G}$ is correct

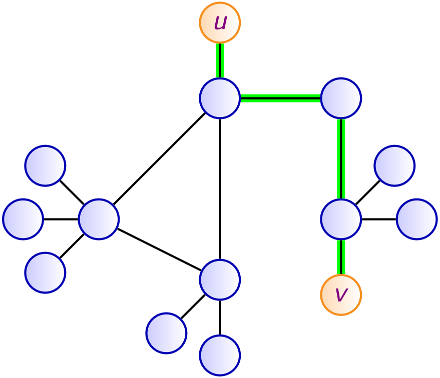
Verify edges: Query each pair in $\hat{E}$.

bounded degree $\implies$ $O(n)$ queries



Question: How do we verify $(a, b) \notin \hat{E}$ is a non-edge?

How to verify non-edges?

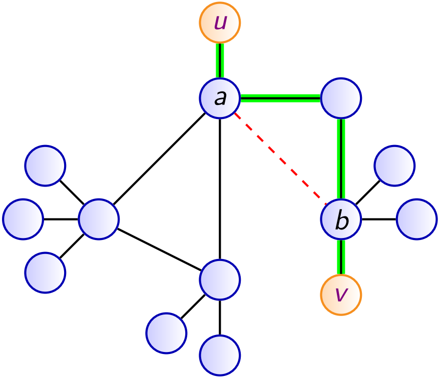


(a, b) is a non-edge if for some $(u, v) \in V^2$, $d(u, v) = \hat{d}(u, v)$ and

$$\hat{d}(u, a) + 1 + \hat{d}(b, v) < \hat{d}(u, v).$$

For $(u, v) \in V^2$, let $S_{u,v} = \left\{ (a, b) \notin \hat{E} \text{ with this property} \right\}$.

How to verify non-edges?

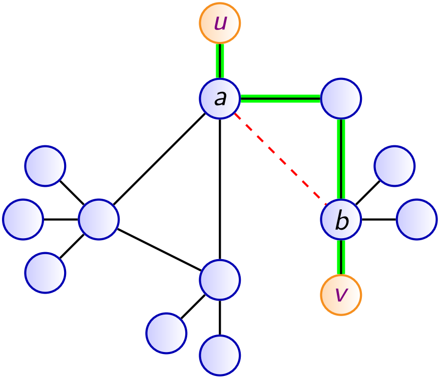


(a, b) is a non-edge if for some $(u, v) \in V^2$, $d(u, v) = \hat{d}(u, v)$ and

$$\hat{d}(u, a) + 1 + \hat{d}(b, v) < \hat{d}(u, v).$$

For $(u, v) \in V^2$, let $S_{u,v} = \left\{ (a, b) \notin \hat{E} \text{ with this property} \right\}$.

How to verify non-edges?



Greedy

(a, b) is a non-edge if for some $(u, v) \in V^2$, $d(u, v) = \hat{d}(u, v)$ and

$$\hat{d}(u, a) + 1 + \hat{d}(b, v) < \hat{d}(u, v).$$

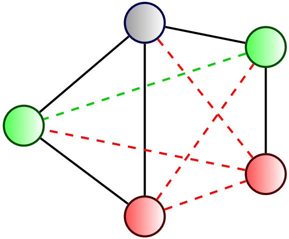
For $(u, v) \in V^2$, let $S_{u,v} = \left\{(a, b) \notin \hat{E} \text{ with this property}\right\}$.

Reduction to Set-Cover

Set-Cover instance:

universe: all the non-edges of $\hat{G}$

sets: $S_{u,v}$ for every pair (u, v)



Greedy Set-Cover uses $O(\log n) \cdot OPT$ sets

Greedy non-edge verification uses $O(\log n) \cdot OPT$ queries

Bounding OPT by $n^{1+o(1)}$

Another algorithm for non-edge verification:

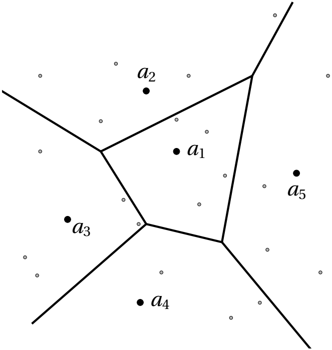
1. Voronoi cell decomposition $\Longrightarrow \tilde{O}(n\sqrt{n})$
2. Recursion $\Longrightarrow n^{1+o(1)}$

Greedy is simpler

Warm-up: bounding OPT by $\tilde{O}(n\sqrt{n})$

A : set of $\sqrt{n}$ centers

$Voronoi(a)$: set of vertices closer to a than to $A \backslash \{a\}$



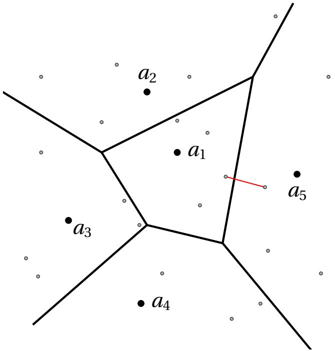
Incorrect algorithm

1. Pick a subset A of V
2. Compute $Voronoi(a)$ for all $a \in A$
3. For every $a \in A$, verify $G[Voronoi(a)]$ by exhaustive queries

Warm-up: bounding OPT by $\tilde{O}(n\sqrt{n})$

A : set of $\sqrt{n}$ centers

$Voronoi(a)$: set of vertices closer to a than to $A \setminus \{a\}$



Incorrect algorithm

1. Pick a subset A of V
2. Compute $Voronoi(a)$ for all $a \in A$
3. For every $a \in A$, verify $G[Voronoi(a)]$ by exhaustive queries

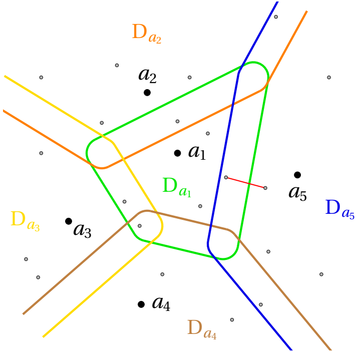
Warm-up: bounding OPT by $\tilde{O}(n\sqrt{n})$

A : set of $\sqrt{n}$ centers

D_a : subgraph associated to $a \in A$, a bit larger than $Voronoi(a)$

Goal :

1. $\bigcup_A G[D_a]$ covers E
2. $|D_a| = O(\sqrt{n})$



Correct algorithm

1. Pick a subset A of V
2. Compute D_a for all $a \in A$
3. For every $a \in A$, verify $G[D_a]$ by exhaustive queries

Definition of D_a

Goal :

1. $\bigcup_A G[D_a]$ covers E
2. $|D_a| = O(\sqrt{n})$

Notation: $C(b) = \{\text{vertices in } b\text{'s Voronoi cell, if } b \text{ was added to } A\}$

Definition: $D_a = \bigcup \{C(b) : d(a,b) \le 2\}$

Lemma [MZ 2013]: $\bigcup_A G[D_a]$ covers E.

Observation: If every $C(b)$ has size $< \sqrt{n}$, then $|D_a| = O(\sqrt{n})$ for all a.

How to compute A and $\{D_a\}$?

Idea from compact routing:

Lemma (Thorup Zwick 2001)

There exists a set A of size $O(\sqrt{n} \cdot \log n)$ such that every $C(b)$ has size $O(\sqrt{n})$. It can be computed in polynomial time when the graph is given.

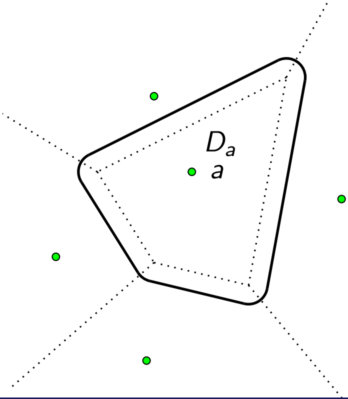
Our approach:

1. In the guessed graph $\hat{G}$, compute A and $\{C(b)\}$, thus obtaining $\{D_a\}$.
2. Check whether $\{D_a\}$ in G and in $\hat{G}$ are the same: $\tilde{O}(n\sqrt{n})$ queries.

Extension: bounding OPT by $n^{1+o(1)}$

Algorithm

1. Pick a subset A of V
2. Compute D_a for all $a \in A$
3. For every $a \in A$, verify $G[D_a]$ by ~~exhaustive queries~~ recursion

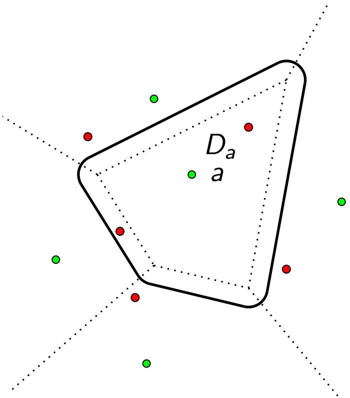


1. Allow selection of centers *outside* the cell.
2. Limit the subcells to being contained *inside* the cell.

Extension: bounding OPT by $n^{1+o(1)}$

Algorithm

1. Pick a subset A of V
2. Compute D_a for all $a \in A$
3. For every $a \in A$, verify $G[D_a]$ by ~~exhaustive queries~~ recursion

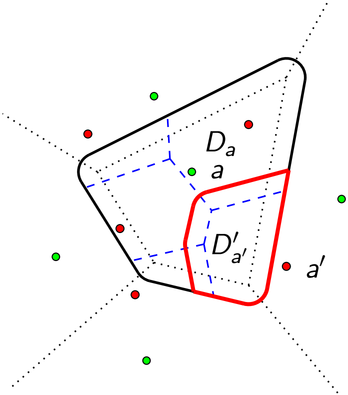


1. Allow selection of centers *outside* the cell.
2. Limit the subcells to being contained *inside* the cell.

Extension: bounding OPT by $n^{1+o(1)}$

Algorithm

1. Pick a subset A of V
2. Compute D_a for all $a \in A$
3. For every $a \in A$, verify $G[D_a]$ by ~~exhaustive queries~~ recursion



1. Allow selection of centers *outside* the cell.
2. Limit the subcells to being contained *inside* the cell.

Graphs of bounded degree

Reconstruction using distance queries [MZ 2013]

- $\tilde{O}(n\sqrt{n})$ algorithm using Voronoi cell decomposition

Verification

- $n^{1+o(1)}$ greedy algorithm

Reconstruction using shortest path queries

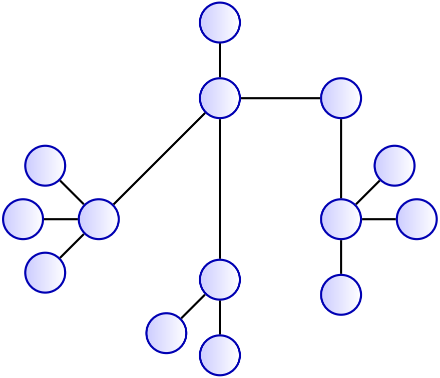
- $n^{1+o(1)}$ greedy algorithm

Open question

Is there a near-linear algorithm for reconstruction using distance queries?

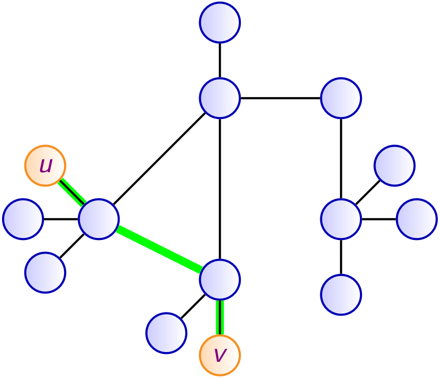
Reconstruction using shortest path queries

Suppose we know a subgraph H of G.



Reconstruction using shortest path queries

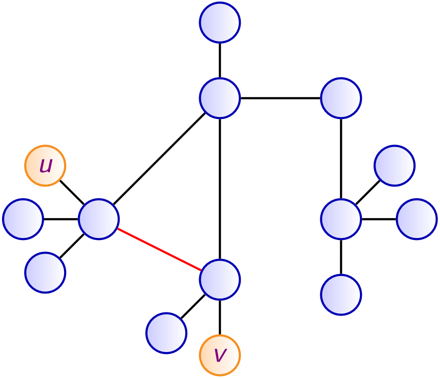
Suppose we know a subgraph H of G.



$$d_G(u, v) \neq d_H(u, v)$$

Reconstruction using shortest path queries

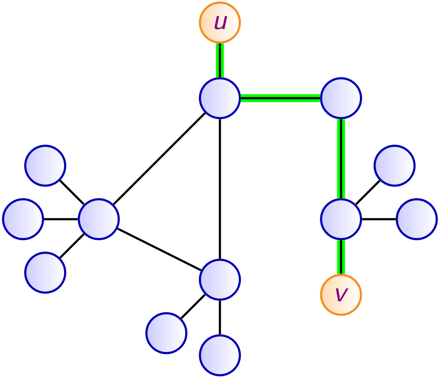
Suppose we know a subgraph H of G.



find an edge of G and update H

Reconstruction using shortest path queries

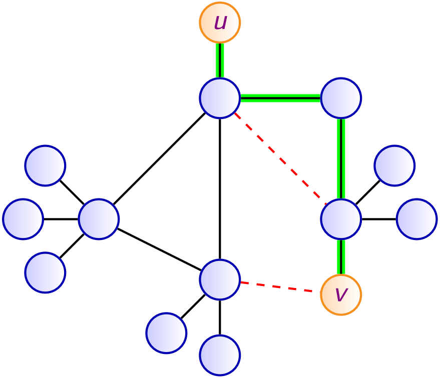
Suppose we know a subgraph H of G.



$$d_G(u, v) = d_H(u, v)$$

Reconstruction using shortest path queries

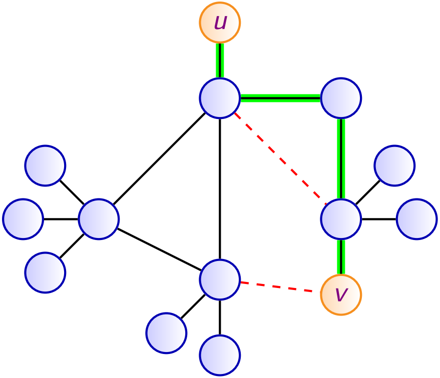
Suppose we know a subgraph H of G.



confirm non-edges of G

Reconstruction using shortest path queries

Suppose we know a subgraph H of G.



Greedy

confirm non-edges of G

Reconstruction using shortest path queries

- $\#\big(\text{queries such that } d_G(u,v) \neq d_H(u,v)\big) = O(n)$.
- $\#\big(\text{queries such that } d_G(u,v) = d_H(u,v)\big) = O(\log n) \cdot OPT$, where $OPT = n^{1+o(1)}$ is the optimum number of queries for verification.

Overall query complexity: $n^{1+o(1)}$

Graphs of bounded degree

Reconstruction using distance queries [MZ 2013]

- $\tilde{O}(n\sqrt{n})$ algorithm using Voronoi cell decomposition

Verification

- $n^{1+o(1)}$ greedy algorithm

Reconstruction using shortest path queries

- $n^{1+o(1)}$ greedy algorithm

Open question

Is there a near-linear algorithm for reconstruction using distance queries?

Thank you!

Attempting a 3-level algorithm for reconstruction

1. Pick A of size $\tilde{O}(n^{1/3})$
2. Query $(a, v)\ \forall (a, v) \in A \times V$, and obtain cells $\{D_a\}_{a \in A}$
3. In each cell D_a
4. Pick $A' \subseteq D_a$ of size $\tilde{O}(n^{1/3})$
5. Query $(a', v)\ \forall (a', v) \in A' \times D_a$, and obtain subcells $\{D'_{a'}\}_{a' \in A'}$
6. For each subcell $D'_{a'}$, reconstruct $G[D'_{a'}]$ by exhaustive queries.

We might hope it's a $\tilde{O}(n^{4/3})$ algorithm.

What goes wrong for recursion?

To bound the size of each cell, earlier we could write

$$D_a = \bigcup \{ C(b) : d(a, b) \leq 2 \}.$$

Now we can write

$$D'_{a'} = \bigcup \{ C'(b) : d(a', b) \leq 2 \}.$$

But b may be outside D_a. Problem!