Network Discovery and Landmarks in Graphs

Thomas Erlebach

Joint work with:
Zuzana Beerliova, Felix Eberhard,
Alexander Hall, Shankar Ram (ETH Zurich),
Matúš Mihaľák, Michael Hoffmann (Leicester)
Discover information about an unknown network using queries.

Verify information about a network using queries.

“Network” means connected, undirected graph.
**Network Discovery:**

- Only the set $V$ of network nodes is known in the beginning.
- Task: Identify all edges and non-edges of the network using a small number of queries.
- On-line problem, competitive analysis

**Network Verification:**

- Check whether an existing network “map” is correct using a small number of queries.
- Off-line problem, approximation algorithms
Simple Theoretical Model

The LG-Model (LG = Layered Graph):

- Connected graph $G = (V, E)$ with $|V| = n$ (in the on-line case, only $V$ is known)

- Query at node $v \in V$ yields the subgraph containing all shortest paths from $v$ to all other nodes in $G$.

- Problem LG-ALL-DISCOVERY (LG-ALL-VERIFICATION):
  Minimize the number of queries required to discover (verify) all edges and non-edges of $G$.

(motivated by Internet AS graph discovery)
Simple Theoretical Model

The LG-Model (LG = Layered Graph):

- Connected graph $G = (V, E)$ with $|V| = n$ (in the on-line case, only $V$ is known)
- Query at node $v \in V$ yields the subgraph containing all shortest paths from $v$ to all other nodes in $G$.
- Problem LG-ALL-DISCOVERY (LG-ALL-VERIFICATON): Minimize the number of queries required to discover (verify) all edges and non-edges of $G$.

(motivated by Internet AS graph discovery)

Observation. Query at $v$ discovers all edges and non-edges between vertices with different distance from $v$. 
Example
Example
Example
Example
Example
Example
Example
Overview of Results

**Network Discovery**

- No deterministic algorithm can be better than $3$-competitive
- $O(\sqrt{n \log n})$-competitive randomised algorithm

**Network Verification**

- Equivalent to placing ‘landmarks’ in graphs
- $o(\log n)$-inapproximability result
- $\Theta(d/\log d)$ queries suffice for hypercubes
Network Discovery
Competitive Ratio

An algorithm for LG-ALL-Discovery is $\rho$-competitive (has competitive ratio $\rho$) if, on any input graph $G$, the number of queries the algorithm makes is at most $\rho$ times as large as the optimal number of queries for that graph.

A randomised algorithm for LG-ALL-Discovery is $\rho$-competitive (has competitive ratio $\rho$) if, on any input graph $G$, the expected number of queries the algorithm makes is at most $\rho$ times as large as the optimal number of queries for that graph.
Competitive Lower Bounds
Competitive Lower Bounds

Optimal number of queries: 4
Competitive Lower Bounds

Any deterministic on-line algorithm:
Competitive Lower Bounds

Any deterministic on-line algorithm:
Any deterministic on-line algorithm:
Competitive Lower Bounds

Any deterministic on-line algorithm:
Competitive Lower Bounds

Any deterministic on-line algorithm:

Needs at least 9 queries!

In general: \( \text{OPT} = k \), \( \text{ALG} = 2k + 1 \)

\( \Rightarrow \) No det. algorithm can be better than 2-competitive.
Competitive Lower Bounds

Any deterministic on-line algorithm:

\[ \text{OPT} = k, \quad \text{ALG} = 2k + 1 \]

\[ \Rightarrow \text{No det. algorithm can be better than } 2\text{-competitive.} \]

Also: No rand. algorithm can be better than \( \frac{4}{3} \)-competitive.
Improved Lower Bound

Construction of improved deterministic lower bound 3:
On-Line Algorithm

Every (non-)edge can either be discovered by many (more than $T$) queries or by few (at most $T$) queries.
On-Line Algorithm

1. Every (non-)edge can either be discovered by many (more than $T$) queries or by few (at most $T$) queries.
2. Phase 1: Use random queries to discover all (non-)edges that can be discovered by many queries.

Let $T = p n \ln n$ and make $3T$ queries in Phase 1, we obtain competitive ratio $O(p n \log n)$. 

---

On-Line Algorithm

- Every (non-)edge can either be discovered by many (more than $T$) queries or by few (at most $T$) queries.
- Phase 1: Use random queries to discover all (non-)edges that can be discovered by many queries.
- Phase 2: For each remaining undiscovered (non-)edge, query all vertices that discover it.

By choosing $T = p n \ln n$ and making $3T$ queries in Phase 1, we obtain competitive ratio $O(\frac{p n \log n}{})$. 
On-Line Algorithm

- Every (non-)edge can either be discovered by many (more than $T$) queries or by few (at most $T$) queries.
- Phase 1: Use random queries to discover all (non-)edges that can be discovered by many queries.
- Phase 2: For each remaining undiscovered (non-)edge, query all vertices that discover it.
- By choosing $T = \sqrt{n \ln n}$ and making $3T$ queries in Phase 1, we obtain competitive ratio $O(\sqrt{n \log n})$. 
Network Verification
Network Verification

Given a connected graph $G = (V, E)$, find a smallest set $Q \subset V$ such that the queries at $Q$ verify all edges and non-edges.
Network Verification

Given a connected graph $G = (V, E)$, find a smallest set $Q \subset V$ such that the queries at $Q$ verify all edges and non-edges.

$Q$ must be such that for every two nodes $u, v \in V$, $u \neq v$, there is at least one vertex in $Q$ with different distance from $u$ and $v$. 
Network Verification

Given a connected graph $G = (V, E)$, find a smallest set $Q \subset V$ such that the queries at $Q$ verify all edges and non-edges.

$Q$ must be such that for every two nodes $u, v \in V$, $u \neq v$, there is at least one vertex in $Q$ with different distance from $u$ and $v$.

Equivalently, $Q$ must be such that every vertex in $V$ is uniquely identified by the vector of its distances from the vertices in $Q$. 
Network Verification

- Given a connected graph \( G = (V, E) \), find a smallest set \( Q \subseteq V \) such that the queries at \( Q \) verify all edges and non-edges.
- \( Q \) must be such that for every two nodes \( u, v \in V, u \neq v \), there is at least one vertex in \( Q \) with different distance from \( u \) and \( v \).
- Equivalently, \( Q \) must be such that every vertex in \( V \) is uniquely identified by the vector of its distances from the vertices in \( Q \).
- Problem has been studied as placing landmarks in graphs (also: metric dimension)
Known Results for Metric Dimension

- Khuller, Raghavachari, Rosenfeld 1996:
  - NP-hard, $O(\log n)$-approximation
  - **Paths**: dimension 1
  - **Trees**: dimension $\sum_{v: \ell_v > 1}(\ell_v - 1)$, where $\ell_v$ is the number of legs at $v$
  - **Claim**: $d$-dimensional grids have metric dimension $d$. 

---

Known Results for Metric Dimension

- Khuller, Raghavachari, Rosenfeld 1996:
  - NP-hard, $O(\log n)$-approximation
  - **Paths:** dimension 1
  - **Trees:** dimension $\sum_{v: \ell_v > 1}(\ell_v - 1)$, where $\ell_v$ is the number of legs at $v$
  - **Claim:** $d$-dimensional grids have metric dimension $d$
    We can show that this is wrong at least for hypercubes!
Theorem. There is no $o(\log n)$ approximation algorithm for LG-ALL-VERIFICATION unless $P = NP$.

Proof Idea: Reduction from Test Collection Problem
Theorem. The optimal solution to LG-ALL-VALIDATION in $d$-dimensional hypercubes has size $\Theta(d/ \log d)$.

Proof Idea:

- Lower bound of $\Omega(d/ \log d)$ follows from general lower bound $\log_{d+1} n$ for graphs of diameter $d$.
- Upper bound: choose $O(d/ \log d)$ queries uniformly at random, and show that with positive probability they verify all edges and non-edges.
Conclusion

Network discovery and verification in the LG model.

**Discovery:** $O(\sqrt{n \log n})$-competitive algorithm, lower bounds of 3 (deterministic) and $\frac{4}{3}$ (randomised)

**Verification:** $\Theta(\log n)$-approximable, $\Theta(d/ \log d)$ queries for $d$-dimensional hypercubes

**Ongoing work:**

- Optimal query sets for specific graph classes.
- Experimental work on greedy-type algorithms.
- Extension to other models (shortest-path tree, only distances, . . .) or discovery goals (e.g., determine network diameter).
Thank you!