Complexity vs. Optimality: Unraveling Source-Destination Connection in Uncertain Graphs

Xinzhe Fu, Zhiying Xu, Qianyang Peng, Luoyi Fu and Xinbing Wang

April 27, 2017
1 Motivations

2 Problem Formulation
   ■ Modeling
   ■ Problem Definition

3 Computational Complexity

4 Proposed Algorithms
   ■ Exact Algorithm
   ■ Approximation Algorithms

5 Experiments
Outline

1 Motivations

2 Problem Formulation
   ■ Modeling
   ■ Problem Definition

3 Computational Complexity

4 Proposed Algorithms
   ■ Exact Algorithm
   ■ Approximation Algorithms

5 Experiments
Motivating Examples

Can the two nodes communicate with each other?
Motivating Examples

Is the two papers related with each other?
Motivating Examples

- Link Failure, “Gift Citation”
- Solution: Link Probing, Text Mining

Communication Network

Citation Network
Motivating Examples

- **Communication Network**
- **Citation Network**

- **Link Failure, “Gift Citation”**
- **Solution:** Link Probing, Text Mining
Uncertainty is Prevalent

Tradition Deterministic Graph Is Not Enough!

- Well-studied problem
- Graph Reachability

- Injecting Uncertainty into Graphs
- Redefine the Source-Destination Connectivity Determination
Outline

1 Motivations

2 Problem Formulation
   ■ Modeling
   ■ Problem Definition

3 Computational Complexity

4 Proposed Algorithms
   ■ Exact Algorithm
   ■ Approximation Algorithms

5 Experiments
Uncertain Graph

- Network topology (Prior)
- Prior existence probability (Edges)
- Testing cost (Edges)
- Composition of realizations

Uncertain Graph (Edge Uncertainty)
For an uncertain graph $G$, denote $G$ as its underlying realization. $G$ can be interpreted as a product distribution over all its possible realizations.
The key elements of an uncertain graph $G(V, E, p, c)$:

- $V$: vertex set
- $E$: edge set
- $p$: $E \mapsto (0, 1]$, probability function
- $c$: $E \mapsto \mathbb{R}^+$, cost function
Problem Definition

Definition (Problem Formulation)

Given an uncertain graph $G(V, E, p, c)$ and two nodes $s, t \in V$ designated as source and destination, find a testing strategy to determine the $s$-$t$ connectivity while incurring the minimum expected cost.

- The results of tests are dictated by the (priorly unknown) underlying graph.
- The expectation of cost is taken over all possible realizations of $G$.
- The testing strategy can be adaptive.
Problem Definition

Definition (Problem Formulation)

Given an uncertain graph $G(V, E, p, c)$ and two nodes $s, t \in V$ designated as source and destination, find a testing strategy to determine the $s$-$t$ connectivity while incurring the minimum expected cost.

- The results of tests are dictated by the (priorly unknown) underlying graph.
- The expectation of cost is taken over all possible realizations of $G$.
- The testing strategy can be adaptive.
Adaptive Testing Strategy

How to properly define the strategy?

- A strategy decide the next edge to test based on the previous results.
- A strategy terminates by verifying the existence of an $s$-$t$ path or an $s$-$t$ cut.

Definition (Temporary State)

A temporary state $s$ of an uncertain graph $G(V, E, p, c)$ is an $|E|$-dimension vector with elements “0”, “1” and “*”. Define $S = \{0, 1, *\}^{|E|}$ to be the set of temporary states associated with $G$. 
Adaptive Testing Strategy

How to properly define the strategy?

- A strategy decides the next edge to test based on the previous results.
- A strategy terminates by verifying the existence of an $s$-$t$ path or an $s$-$t$ cut.

Definition (Temporary State)

A temporary state $s$ of an uncertain graph $G(V, E, p, c)$ is an $|E|$-dimension vector with elements “0”, “1” and “*”. Define $S = \{0, 1, *\}^{|E|}$ to be the set of temporary states associated with $G$. 
Adaptive Testing Strategy

Definition (Temporary State)

A temporary state $s$ of an uncertain graph $G(V, E, p, c)$ is an $|E|$-dimension vector with elements “0”, “1” and “*”. Define $S = \{0, 1, *\}^{|E|}$ to be the set of temporary states associated with $G$.

Definition (Adaptive Testing Strategy)

An adaptive testing strategy is a mapping $\pi : S \mapsto E \cup \{\bot\}$.

- Define $E_\pi(G)$ as the set of edges strategy $\pi$ on the underlying graph $G$.
- The expected cost of $\pi$ is given as $Cost(\pi) = \sum_{G \in \mathcal{G}}[Pr(G)\sum_{e \in E_\pi(G)}c(e)]$. 
An Example

Definition (Adaptive Testing Strategy)
An adaptive testing strategy is a mapping $\pi: S \mapsto E \cup \{\perp\}$. 

![Diagram showing an example of adaptive testing strategy]

- Testing Strategy
  - Values: $\{\ast, \ast, \ast\}, \{0, \ast, \ast\}, \{0, \ast, 1\}, \{\ast, \ast, 1\}, \{\ast, \ast, 0\}, \{\ast, 0, \ast\}, \{\ast, 0, 1\}, \{\ast, 0, 0\} 
  - Edges: $e_1, e_2, e_3, e_4, e_5$

- Uncertain graph
  - Edges: $e_1, e_2, e_3$
  - Probabilities: $p(e_1) = 0.2, p(e_2) = 0.5, p(e_3) = 0.6$

- A possible underlying graph
  - Edges: $e_1, e_2, e_3$
  - Costs: $c(e_1) = 4, c(e_2) = 5, c(e_3) = 2$

- Evolution of Temporary State
  - Graph transitions 
  - Known edges: $e_1, e_2, e_3$
  - Known non-edge: $e_4$
  - Potential edge: $e_5$
Outline

1. Motivations

2. Problem Formulation
   - Modeling
   - Problem Definition

3. Computational Complexity

4. Proposed Algorithms
   - Exact Algorithm
   - Approximation Algorithms

5. Experiments
Two Variants of the Problem

1. Compute the whole strategy (P1)

Definition (Decision Version of P1)

Given an uncertain graph $G(V, E, p, c)$ and two nodes $s, t \in V$ designated as source and destination, is there a testing strategy that determines the $s$-$t$ connectivity with expected cost less than $k$.

2. Compute the strategy sequentially (P2)

Definition (Decision Version of P2)

Given an uncertain graph $G(V, E, p, c)$, two nodes $s, t \in V$ designated as source and destination and the current temporary state, decide the optimal next edge to test.
Two Variants of the Problem

1. Compute the whole strategy (P1)

**Definition (Decision Version of P1)**

Given an uncertain graph $G(V, E, p, c)$ and two nodes $s, t \in V$ designated as source and destination, is there a testing strategy that determines the $s$-$t$ connectivity with expected cost less than $k$.

2. Compute the strategy sequentially (P2)

**Definition (Decision Version of P2)**

Given an uncertain graph $G(V, E, p, c)$, two nodes $s, t \in V$ designated as source and destination and the current temporary state, decide the optimal next edge to test.
Complexity-theoretic Results on P1

Theorem

Computing the expected cost of the optimal strategy is \#P-hard. (The decision version of a \#P-hard problem is NP-hard)

Proof

- By reduction from network (s-t) reliability problem.

---

1L. G. Valiant, “The complexity of enumeration and reliability problems”
Complexity-theoretic Results on P2

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding the optimal first edge to test is NP-hard.</td>
</tr>
</tbody>
</table>

| Proof. |
| The proof is done by reduction from the set cover problem. |

| Definition (The Set Cover Problem) |
| Given a universe \( \mathcal{U} \) of elements, a family \( \mathcal{S} \) of subsets of the universe and a predefined integer \( k \), does there exist a subfamily \( \mathcal{C} \subseteq \mathcal{S} \) such that \( \bigcup_{C \in \mathcal{C}} = \mathcal{U} \) and \( |\mathcal{C}| \leq k \). |
Theorem

Deciding the optimal first edge to test is NP-hard.

Proof.

Through appropriately assigning the value of $P_s$, $C_s$, $P_M$, $C_M$, $P_e$, $C_e$, we can show that the optimal first edge to test is $M$ if and only if there does not exist a set cover $C$ with $|C| \leq k$. □
Outline

1 Motivations

2 Problem Formulation
   ■ Modeling
   ■ Problem Definition

3 Computational Complexity

4 Proposed Algorithms
   ■ Exact Algorithm
   ■ Approximation Algorithms

5 Experiments
Markov Decision Process Framework

**Definition (Markov Decision Process)**

A mathematical model for modeling decision making under uncertain situations. A Markov Decision Process (MDP) model contains:

- State Space
- Decision Epochs
- Action Sets
- Transition Probabilities and Rewards
- Strategy (with certain expected total reward)

Markov Property: the effects of an action only depends on the current state and the action itself.
Mapping the Problem into MDP

The correspondence between the key elements:

- **State Space**: the set of temporary states
  - \( S = S_0 \cup S_1 \cup \ldots \cup S_{|E|} \)
- **Decision Epochs**
- **Action Sets**: testing edges
  - \( A_s, A = \bigcup_{s \in S} A_s = E \cup \{ \perp \} \)
- **Transition Probabilities and Rewards**
  - From \( s \) to \( s \cdot e \)
  - From \( s \) to \( s \setminus e \)
- **Strategy**
Algorithm 1 The MDP-based Exact Algorithm

Input: Uncertain graph $G(V, E, p, c)$, source $s$, destination $t$
Output: The optimal testing strategy $\pi$

1: Initialize: $u_\pi(s) = 0$, for all $s \in S|E|$
2: for $i = |E|$ to 0 do
3:   for All $s$ in $S_i$ do
4:     if $s$ is a terminating state then
5:       $u_\pi(s) := 0$, $\pi(s) := \bot$.
6:     else
7:       $e^* := \arg\max_{e \in A_s} \{-c(e) + p(e)u_\pi(s \cdot e) \newline + (1 - p(e))u_\pi(s \setminus e)\}$,
8:       $u_\pi(s) := -c(e^*) + p(e^*)u_\pi(s \cdot e^*) \newline + (1 - p(e^*))u_\pi(s \setminus e^*)$,  
9:       $\pi(s) := e^*$.
10: return $\pi$
A Simple Greedy Approach

A strategy that tests the edges following the ascending order of costs is an $O(|E|)$ approximation.

**Theorem**

*Given an uncertain graph $G(V, E, p, c)$ and two nodes $s$ and $t$ as source and destination, let $\pi$ be a strategy that tests the edges in $E$ according to their costs sorted in an increasing order. Then, $\text{Cost}(\pi) \leq |E| \cdot \text{Cost}(\pi^*)$, where $\pi^*$ is the optimal strategy.*
Adaptive Submodular Algorithm – Preliminaries

**Definition (Extension)**

For two temporary states \( a, b \in S \), we say \( a \) is an extension of \( b \), written as \( a \sim b \) if \( a_i = b_i \) for all \( b_i \neq * \).

**Definition (Function on Temporary States)**

Let \( g : S \mapsto \mathbb{N} \) be a utility function on temporary states.

- \( g \) is monotonically increasing if \( g(s') - g(s) \geq 0 \) for all \( s \in S, s' \sim s \).

- \( g \) is adaptive submodular if
  
  \[
  g(s \cdot e) - g(s) \geq g(s' \cdot e) - g(s') \quad \text{and} \quad g(s \setminus e) - g(s) \geq g(s' \setminus e) - g(s') \quad \text{whenever} \quad s' \sim s \quad \text{and} \quad s_e = s'_e = *.
  \]
The utility function $g$ should be assignment feasible in the sense that:

- $g(\ast, \ast, \ldots, \ast) = 0$.
- $g(s) = Q$ iff $s$ is a terminating state, where $Q$ is the target value.

**Lemma (The Adaptive Submodular Framework \(^2\))**

*Each time choosing(testing) the edge with the maximum expected gain:*

$$
\frac{p(e)g(s \cdot e) + (1 - p(e))g(s \setminus e) - g(s)}{c(e)}
$$

*yields an $O(\ln |Q|)$-approximation.*

Adaptive Submodular Algorithm – Utility Function

Definition (The Design of Utility Function $g$)

Define $\mathcal{P}$ and $\mathcal{C}$ as the collection of $s$-$t$ paths and $s$-$t$ cuts in $\mathcal{G}$ respectively. Define $\mathcal{P}_e$ and $\mathcal{C}_e$ as the collection of $s$-$t$ paths and $s$-$t$ cuts that edge $e$ lies on in $\mathcal{G}$ respectively. We have,

$$
g_p(s) = \left| \bigcup_{e:s_e=0} \mathcal{P}_e \right|, \quad g_c(s) = \left| \bigcup_{e:s_e=1} \mathcal{C}_e \right|,
$$

$$
g(s) = |\mathcal{P}| |\mathcal{C}| - (|\mathcal{P}| - g_p(s))(|\mathcal{C}| - g_c(s)).
$$

Example

- Set of paths
- Set of cuts
Determine the strategy sequentially

Algorithm 2 The Adaptive Submodular Algorithm

\[ \textbf{Input:} \text{ Uncertain graph } \mathcal{G}(V, E, p, c), \text{ source and destination.} \]

\[ \textbf{Output:} \text{ Testing strategy } \pi \]

1. \textbf{Initialize:} Current state } s := (*, *, \ldots, *), \text{ The set of tested edges } E_\pi \text{ as an empty set.}
2. \textbf{Repeat} until } s \text{ becomes a terminating state.
3. \quad e^* := \arg \max_{e \in E \setminus E_\pi \setminus \{e^*\}} \left\{ \frac{p(e)g(s \setminus e) + (1 - p(e))g(s \setminus e - g(s))}{c(e)} \right\}.
4. \quad E_\pi := E_\pi \cup \{e^*\}, \text{ test } e^* \text{ and observe the outcome.}
5. \quad \textbf{if} edge } e^* \text{ exists \textbf{then}
6. \quad \quad s_{e^*} := 1
7. \quad \textbf{else}
8. \quad \quad s_{e^*} := 0

Test the edge with the maximum marginal gain
Adaptive Submodular Framework – Performance

**Theorem**

\( g \) is monotonically increasing, adaptive submodular, and assignment feasible.

**Proof:**

By the definition of \( g \).

**Theorem**

The Adaptive Submodular Algorithm yields an \( O(\ln |Q|) = O(\ln |P||C|) \)-approximation.

**Proof:**

By the Adaptive Submodular framework proposed by Golovin and Krause \(^3\).

\(^3\)D. Golovin and A. Krause, “Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization”
Outline

1. Motivations
2. Problem Formulation
   - Modeling
   - Problem Definition
3. Computational Complexity
4. Proposed Algorithms
   - Exact Algorithm
   - Approximation Algorithms
5. Experiments
Experiment Settings

Experiment Datasets:
- Citation Networks (273751 nodes, 993025 edges)
- Internet Peer to Peer Networks (5000 nodes, 16469 edges)
- Twitter Ego Networks (213 nodes, 17930 edges)

Parameters Assignments:
- Costs: drawn from $\mathcal{N}(50,100)$
- Probabilities: $p(e = (x, y)) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$

Performance Metric:
- The expected costs are approximated by the averaged costs on 1000 underlying graphs.

$^4\Gamma(x)$ denotes the set of neighbors of $x$. 
Algorithms Involved in Comparisons

- Greedy Algorithm (Greedy)
- Adaptive Submodular Algorithm (AdaSub)
- Optimistic Sort Algorithm (OpSort): following an increasing order of $c/p$
- Pessimistic Sort Algorithm (PeSort): following an increasing order of $c/(1 - p)$
- Intersection Sort Algorithm (IntSort): tests the edge with the minimum cost that lies on the intersection of a shortest s-t path and a minimum s-t cut.
- MDP-based Algorithm (MDP): only on a sequence of subnetworks with 20 edges.
Experiment Results
Thank You!