Scalable Diffusion-Aware Optimization of Network Topology

Elias Khalil
Joint work with Bistra Dilkina, Le Song

School of Computational Science & Engineering
Diffusion on Networks: Why Care?
Some important questions

• Modeling cascading behavior: [Kempe ‘03]
• Source Selection for Influence Maximization:
  • Problem formulation: [Domingos ‘02]
  • Submodularity: [Kempe ‘03]
  • Hardness of influence estimation: [Chen ‘10]
  • Fast heuristics: [Chen ‘10]
• Optimizing the Structure of Diffusion Networks
Optimizing Network Structure
Optimizing Network Structure

Deleting Edges to Minimize a Spread
Optimizing Network Structure

Deleting Edges to Minimize a Spread
Optimizing Network Structure
Optimizing Network Structure

Adding Edges to Maximize a Spread
Optimizing Network Structure

Adding Edges to Maximize a Spread
Optimizing Network Structure

How to strategically modify networks to optimize their susceptibility to cascades?
# State-of-the-art in Optimizing Diffusion Networks

<table>
<thead>
<tr>
<th>Problems</th>
<th>SIR</th>
<th>IC</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ’12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ’12]</td>
<td>[Sheldon ’10]</td>
<td></td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ’10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problems</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guarantees</td>
<td>Eigenvalue-based Approx. guarantees Scalable method</td>
</tr>
<tr>
<td>AND Scalability</td>
<td>Mixed Integer Program Approx. guarantees Not scalable!</td>
</tr>
</tbody>
</table>

KDD, NYC August 26th 2014
Elias Khalil
### State-of-the-art in Optimizing Diffusion Networks

#### MODELS

<table>
<thead>
<tr>
<th>PROBLEMS</th>
<th>SIR</th>
<th>IC</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ‘12]</td>
<td></td>
<td>[Sheldon ‘10]</td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Guarantees** AND **Scalability**

- Eigenvalue-based Approx. guarantees
- Scalable method

- Mixed Integer Program
- Approx. guarantees
- Not scalable!
# State-of-the-art in Optimizing Diffusion Networks

## Problems

<table>
<thead>
<tr>
<th>Edge Deletion</th>
<th>[Tong ‘12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
</tr>
</tbody>
</table>

## Guarantees and Scalability

- **SIR**
  - Eigenvalue-based
  - Approx. guarantees
  - Scalable method

- **IC**
  - Mixed Integer Program
  - Approx. guarantees
  - Not scalable!

**KDD, NYC August 26th 2014**

Elias Khalil
# State-of-the-art in Optimizing Diffusion Networks

<table>
<thead>
<tr>
<th>PROBLEMS</th>
<th>SIR</th>
<th>IC</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong '12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong '12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong '10]</td>
<td></td>
<td>[Sheldon '10]</td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Guarantees AND Scalability**
  - Eigenvalue-based Approx. guarantees
  - Scalable method
  - Mixed Integer Program
  - Approx. guarantees
  - Not scalable!

---

KDD, NYC August 26th 2014

Elias Khalil
# State-of-the-art in Optimizing Diffusion Networks

<table>
<thead>
<tr>
<th>PROBLEMS</th>
<th>SIR</th>
<th>IC</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ‘12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
<td>[Sheldon ‘10]</td>
<td></td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
<td>Mixed Integer Program</td>
<td></td>
</tr>
</tbody>
</table>

- **Guarantees AND Scalability**
  - SIR: Eigenvalue-based Approx. guarantees Scalable method
  - IC: Approx. guarantees Not scalable!

---

KDD, NYC August 26th 2014

Elias Khalil
State-of-the-art in Optimizing Diffusion Networks

<table>
<thead>
<tr>
<th>PROBLEMS</th>
<th>SIR</th>
<th>IC</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ‘12]</td>
<td></td>
<td>[Sheldon ‘10]</td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guarantees AND Scalability</td>
<td>Eigenvalue-based Approx. guarantees Scalable method</td>
<td>Mixed Integer Program Approx. guarantees Not scalable!</td>
<td></td>
</tr>
</tbody>
</table>
State-of-the-art in Optimizing Diffusion Networks

<table>
<thead>
<tr>
<th>PROBLEMS</th>
<th>SIR</th>
<th>IC</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ‘12]</td>
<td>[Sheldon ‘10]</td>
<td></td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Addition</td>
<td></td>
<td>Mixed Integer Program</td>
<td></td>
</tr>
<tr>
<td>Guarantees <strong>AND</strong> Scalability</td>
<td>Eigenvalue-based Approx. guarantees Scalable method</td>
<td>Approx. guarantees Not scalable!</td>
<td></td>
</tr>
</tbody>
</table>

KDD, NYC August 26th 2014
Elias Khalil
### State-of-the-art in Optimizing Diffusion Networks

#### Models

<table>
<thead>
<tr>
<th>SIR</th>
<th>IC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ‘12]</td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
</tr>
<tr>
<td>Node Addition</td>
<td>-</td>
</tr>
<tr>
<td>Guarantees AND Scalability</td>
<td>Eigenvalue-based Approx. guarantees Scalable method</td>
</tr>
<tr>
<td></td>
<td>Mixed Integer Program Approx. guarantees Not scalable!</td>
</tr>
</tbody>
</table>

**This Paper**

KDD, NYC August 26th 2014

Elias Khalil
## State-of-the-art in Optimizing Diffusion Networks

<table>
<thead>
<tr>
<th>Problems</th>
<th>SIR</th>
<th>IC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Deletion</td>
<td>[Tong ‘12]</td>
<td>-</td>
</tr>
<tr>
<td>Edge Addition</td>
<td>[Tong ‘12]</td>
<td>-</td>
</tr>
<tr>
<td>Node Deletion</td>
<td>[Tong ‘10]</td>
<td>[Sheldon ‘10]</td>
</tr>
<tr>
<td>Node Addition</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Guarantees AND Scalability

- Eigenvalue-based Approx. guarantees
- Scalable method

- Mixed Integer Program
- Approx. guarantees
- Not scalable!
Contributions & Outline

1. Theoretical aspects: a new framework for LT
   - New mathematical properties of the model
   - Supermodularity of our problems’ objective functions
   - Near-optimal optimization schemes

2. Algorithmic aspects: designing scalable algorithms
   - Edge Deletion*
     - Specialized Tree Data Structure
   - Edge Addition*
     - Randomized Neighborhood Size Estimation
   - Networks with Millions of edges
     - Linear Time & Space Algorithms
     - Deletion: 10-20% better
     - Addition: 100% better than next best heuristic

*All results trivially extend to node deletion (addition)
Definitions

A possible cascade

\[ G(V, E, w) \]
Definitions

A possible cascade
↔
Live-Edge Graph (L.E.G)

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G)

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G)

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G)

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G)

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G)

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G) 

Space of Live-Edge Graphs 

\[ G(V, E, w) \]

\[ X_G \]
Definitions

\[ G(V, E, w) \]

Space of Live-Edge Graphs

\[ X_G \]

Live-Edge Graph (L.E.G)

\[
\begin{align*}
\Pr[X_1|G] & \quad \Pr[X_2|G] \\
\Pr[X_3|G] & \quad \Pr[X_4|G] \\
\Pr[X_5|G] &
\end{align*}
\]
Definitions

Live-Edge Graph (L.E.G)

Space of Live-Edge Graphs

\[ \mathcal{P}_{\mathcal{X} \mid G} \]

\[ r(a, X_1) = 2 \]

\[ G(V, E, w) \]
Definitions

Live-Edge Graph (L.E.G)

$\text{Pr}[X_1|G] \quad r(a, X_1) = 2$

$\text{Pr}[X_2|G] \quad r(a, X_2)$

$\text{Pr}[X_3|G] \quad r(a, X_3)$

$\text{Pr}[X_4|G] \quad r(a, X_4)$

$\text{Pr}[X_5|G] \quad r(a, X_5)$

Space of Live-Edge Graphs

$X_G$

$G(V, E, w)$

Influence$(a, G)$

$$= \sum_{i=1}^{5} \text{Pr}[X_i|G] \times r(a, X_i)$$

KDD, NYC August 26th 2014

Elias Khalil
Optimization Formulation

• **Edge Deletion**

\[ S^* := \arg\min_{S \subseteq E : |S| = k} \sum_{a \in A} \text{Influence}(a, G \setminus S) \]

\( k \) edges minimizing...

...the sum of the influences of the sources \( A \) on graph \( G(V, E \setminus S, w) \)

• **Edge Addition**

\[ S^* := \arg\max_{S \subseteq C : |S| = k} \sum_{a \in A} \text{Influence}(a, G \cup S) \]

\( A \): set of source nodes \( A \in V \)

\( C \): set of candidates edges for addition \( C \cap E = \emptyset \)
## Challenges in Analyzing the Objectives

### Source Selection objective [Kempe ‘03]

- \((S, G)\) is Monotone w.r.t. node set \(S\) is simple:
- Fixing a L.E.G \(x\):

### Our objectives (e.g. edge deletion)

- \((a, G\backslash S)\) is Monotone w.r.t. edge set \(E\backslash S\) is NOT simple:
- Fixing a source \(a\):
Challenges in Analyzing the Objectives

Source Selection objective [Kempe ‘03]

\(SS, GG)\) is Monotone w.r.t. node set \(A^* S\) is simple:

\[\text{Influence}(S, G)\]

Common to the two terms above

Fixing a L.E.G \(X\):

\[\text{Influence}(S \cup v, G) - \text{Influence}(S, G)\]

\[= \Pr[X | G] \times [r(S \cup v, X) - r(S, X)]\]

\[\geq \]

\[\Rightarrow \text{DONE!}\]

Our objectives (e.g. edge deletion)

\(a, G\backslash S\) is

Monotone w.r.t. edge set \(E\backslash S\) is NOT simple:

Fixing a source \(a\):

\[A^* \equiv \arg\max_{S \subseteq V: |S| = k} \text{Influence}(S, G)\]
Challenges in Analyzing the Objectives

**Source Selection objective [Kempe ‘03]**

- \( SS, GG \) is Monotone w.r.t. node set
- \( A^* \subseteq SS \) is simple:
- \( \sum_{S \subseteq V : |S| = k} \text{Influence}(S, G) \)

**Our objectives (e.g. edge deletion)**

- \( aa, GG \) is Monotone w.r.t. edge set \( EE \)
- \( S^* \) is NOT simple:
- \( \sum_{a \in A} \text{Influence}(a, G \setminus S) \)

1. **Fixing a L.E.G \( X \):**
   - \( \text{Influence}(S \cup v, G) - \text{Influence}(S, G) \)
   - \( = \sum_{x \in X \setminus G \setminus (S \cup e)} \Pr[X \setminus G \setminus (S \cup e)] \times r(a, X) - \sum_{x \in X \setminus G \setminus S} \Pr[X \setminus G \setminus S] \times r(a, X) \)

2. **Fixing a L.E.G \( X \):**
   - \( \geq \)
   - \( \iff \) DONE!

3. **Fixing a source \( a \):**
   - \( \text{Influence}(a \setminus G \setminus (S \cup e)) - \text{Influence}(a, G \setminus S) \)

4. **Fixing a source \( a \):**
   - \( X_{G \setminus (S \cup e)} \neq X_{G \setminus S} \)
Challenges in Analyzing the Objectives

**Source Selection**

Objective [Kempe ‘03]

\( S, G \) is Monotone w.r.t. node set \( S \)

\( S^* \) is simple.

Influence \((S, G)\)

- Fixing a L.E.G \( X \):
  \[
  \text{Influence}(S \cup v, G) - \text{Influence}(S, G) = \Pr[X | G] \times [r(S \cup v, X) - r(S, X)]
  \]

Common to the two terms above

\[ \geq \]

DONE!

- Fixing a source \( a \):
  \[
  \text{Influence}(a, G \setminus S) - \text{Influence}(a, G \setminus S) = \sum_{x \in X_G \setminus (S \cup e)} \Pr[X | G \setminus (S \cup e)] \times r(a, X) - \sum_{x \in X} \Pr[X | G \setminus S] \times r(a, X)
  \]

**Our Objectives**

(e.g. edge deletion)

\( a, G \setminus S \) is Monotone w.r.t. edge set \( E \)

\( S^* \) is NOT simple.

\[ \sum_{a \in A} \text{Influence}(a, G \setminus S) \]

- Fixing a source \( a \):
  \[
  \text{Influence}(a, G \setminus S) - \text{Influence}(a, G \setminus S) = \Pr[X_G \setminus (S \cup e)] \neq X_G \setminus S
  \]

Sums are over different spaces!

Cannot compare summations directly!
Challenges in Analyzing the Objectives

Source Selection objective [Kempe ‘03]

\( SS, GG \) is Monotone w.r.t. node set \( A \setminus S \) is simple:

Our objectives (e.g. edge deletion)

\( aa, GG \setminus S \) is Monotone w.r.t. edge set \( EE \)

\( A^\ast \subseteq S \) is simple:

Fixing a L.E.G. \( X \):

\[
\text{Influence}(S) = \Pr[X|G] \times [r(S \cup v, X) - r(S, X)]
\]

\[
= \sum_{x \in X \setminus G \setminus S(e)} \Pr[X|G \setminus S(e)] \times r(a, X) - \sum_{x \in G \setminus S} \Pr[X|G \setminus S] \times r(a, X)
\]

Common to the two terms above

\[ \geq \]

DONE!

\[
X_{G \setminus (S \cup e)} \neq X_{G \setminus S}
\]

Sums are over different spaces!

Cannot compare summations directly!
A Deeper Understanding of LT

• Four Properties:
  1. Within-Space Mapping
  2. Space Inclusion
  3. Across-Space Mapping
  4. Across-Space Probability Mapping

} How is a space of live-edge graphs structured w.r.t. a given edge?

} How are different spaces related w.r.t. a given edge?
A Deeper Understanding of LT

- Four Properties:
  1. Within-Space Mapping
  2. Space Inclusion
  3. Across-Space Mapping
  4. Across-Space Probability Mapping

Exploiting the 4 properties

⇒ Monotonicity

⇒ Supermodularity
Supermodularity

• “Increasing differences” property:

\[ f(S \cup e) - f(S) \leq f(T \cup e) - f(T) \]

\( \forall S \subseteq T, e \notin T \)
Supermodularity

- "Increasing differences" property:

\[ f(S \cup e) - f(S) \leq f(T \cup e) - f(T) \]
\[ \forall S \subseteq T, e \notin T \]

**Edge Deletion**

- Deleting \((w, x)\) disconnects \(w\) from \(x\) and \(y\)
- Deleting \((w, x)\) disconnects \(w\) from \(x\) ONLY

**Edge Addition**

- Adding \(e\) connects \(w\) to \(y\) ONLY
- Adding \(e\) connects \(w\) to \(y\) AND \(z\)
**Edge Deletion:**

A Greedy Algorithm

\[
S^* := \arg\min_{S \subseteq E: |S| = k} \sum_{a \in A} \text{Influence}(a, G \setminus S)
\]

- Minimizing Supermodular \( f(.) \) \( \iff \) Maximizing Submodular \( h(.) = C - f(.) \)

- Greedy algorithm [Nemhauser ’63]: At each iteration, add to \( S^* \) the edge \( e^* \) whose deletion results in the largest decrease in the objective value.

- Approximation guarantee:

\[
h(S^*) \geq (1 - 1/e - \alpha)h(S^{OPT})
\]

\( \alpha \): approx. factor for influence estimation
Edge Deletion: A Superior Performance

MemeTracker Dataset

Relative Influence w.r.t unmodified network

Influence_{k \text{ edges deleted}} / Influence_{no \text{ edges deleted}}

Lower is Better

GreedyCutting
Random
Weights
Betweenness
Eigen
Degree

KDD, NYC August 26th 2014
Elias Khalil
Edge Deletion: A Superior Performance

MemeTracker Dataset

Relative Influence w.r.t. unmodified network

Lower is Better

Influence_{k \ edges \ deleted} \over \text{Influence}_{no \ edges \ deleted}
Edge Deletion: A Superior Performance

MemeTracker Dataset

Relative Influence w.r.t unmodified network

\[ \frac{\text{Influence}_{k \text{ edges deleted}}}{\text{Influence}_{\text{no edges deleted}}} \]

Lower is Better
Edge Deletion: A Superior Performance

Influence $k_{edges deleted}$

Influence $no_{edges deleted}$

Relative Influence w.r.t unmodified network

Lower is Better

MemeTracker Dataset

Budget (k)

KDD, NYC August 26th 2014

Elias Khalil
Edge Deletion: A Superior Performance

MemeTracker Dataset

Relative Influence w.r.t unmodified network

\[ \frac{\text{Influence}_{k \text{ edges deleted}}}{\text{Influence}_{\text{no edges deleted}}} \]

~2% of edges

Lower is Better

KDD, NYC August 26th 2014

Elias Khalil
Edge Addition: A Modular Approximation

\[ S^* := \arg\max_{S \subseteq C: |S| = k} \sum_{a \in A} \text{Influence}(a, G \cup S) \]

- **Modular approximation** [Iyer '13]:
  For each candidate edge \( e^* \) (independently), compute the increase in the objective when \( e^* \) is added to the network. Choose the top-\( k \) edges.

- **Approximation guarantee:**
  \[ g(S^*) \leq \beta \left( \frac{1}{1 - \kappa_g} \right) g(S^{OPT}) \]

\( \beta \): approx. factor for influence estimation
\( \kappa_g \): curvature of \( g \)
Edge Addition: A Superior Performance

MemeTracker Dataset

Influence Increase relative to unmodified network

Higher is Better

\[ \text{Influence}_{k \text{ edges added}} - \text{Influence}_{\text{no edges added}} \]

KDD, NYC August 26th 2014
Elias Khalil
Edge Addition: A Superior Performance

Influence $\Delta = \frac{\text{Influence}_{k \text{ edges added}} - \text{Influence}_{\text{no edges added}}}{\text{Influence}_{\text{no edges added}}}$

MemeTracker Dataset

Higher is Better

KDD, NYC August 26th 2014
Elias Khalil
Edge Addition: A Superior Performance

MemeTracker Dataset

Influence Increase relative to unmodified network

\[ \frac{\text{Influence}_{k \text{ edges added}} - \text{Influence}_{\text{no edges added}}}{\text{Influence}_{\text{no edges added}}} \]

KDD, NYC August 26th 2014
Elias Khalil
Scaling Up!

• Both algorithms perform well
• However, naïve implementations
  $\rightarrow O(|V|^2)$ or worse time complexity:
  $\rightarrow$ Cannot scale to large networks!
• Solution: exploit problem structure
  $\rightarrow$ linear time and space algorithms!
Scaling Up!

• Both algorithms perform well.
• However, naïve implementations → $O(|V|^2)$ or worse time complexity:
  → Cannot scale to large networks!
• Solution: exploit problem structure → linear time and space algorithms!
Scaling Up!

- Both algorithms perform well.
- However, naïve implementations → $O(|V|^2)$ or worse time complexity:
  → Cannot scale to large networks!
- Solution: exploit problem structure → linear time and space algorithms!

Details in next slides

Details in paper

KDD, NYC August 26th 2014

Elias Khalil
Naïve Edge Deletion

- Greedy algorithm sketch:
  1. Sample live-edge graphs; induce subgraphs rooted at sources
  2. For 1 to Budget
  3. For each induced subgraph
  4. For each edge $e$ in subgraph
  5. Evaluate the impact of deleting edge $e$
  6. Add $e^*$, the edge with largest average impact, to the solution
  7. Delete $e^*$ from $E$

Example subgraph
Naïve Edge Deletion

• Greedy algorithm sketch:
  1. Sample live-edge graphs; induce subgraphs rooted at sources
  2. For 1 to Budget
  3. For each induced subgraph
  4. For each edge \( e \) in subgraph
  5. Evaluate the impact of deleting edge \( e \)
  6. Add \( e^* \), the edge with largest average impact, to the solution
  7. Delete \( e^* \) from \( E \)

\[ O(|V|) \times O(|V|) \]
Naïve Edge Deletion

• Greedy algorithm sketch:
  1. Sample live-edge graphs; induce subgraphs rooted at sources
  2. For 1 to Budget
  3. For each induced subgraph
  4. For each edge $e$ in subgraph
  5. Evaluate the impact of deleting edge $e$
  6. Add $e^*$, the edge with largest average impact, to the solution
  7. Delete $e^*$ from $E$

Quadratic Time Complexity in $|V|$
Naïve Edge Deletion

• Greedy algorithm sketch:
  1. Sample live-edge graphs; induce subgraphs rooted at sources
  2. For 1 to Budget
  3. For each induced subgraph
  4. For each edge $e$ in subgraph
  5. Evaluate the impact of deleting edge $e$
  6. Add $e^*$, the edge with largest average impact, to the solution
  7. Delete $e^*$ from $E$

Quadratic Time Complexity in $|V|$.

How can we avoid this bottleneck?
Scaling up Edge Deletion

• Observation: Subgraphs induced from live-edge graphs are all **TREES**!

• $\text{Score}(\text{edge } (a, b)) = \#\text{descendants of } b + 1$
Scaling up Edge Deletion

• Observation: Subgraphs induced from live-edge graphs are all **TREES**!

• \( \text{Score}(\text{edge } (a, b)) = \#\text{descendants of } b + 1 \)

Stage 1: Top-Down BFS

![Diagram of a tree with nodes labeled a, b, c, d, e, f, and edges labeled (d, f), (c, e), (b, d), (b, c), (a, b)].
Scaling up Edge Deletion

- Observation: Subgraphs induced from live-edge graphs are all **TREES**!
- \( \text{Score}(\text{edge } (a, b)) = \#\text{descendants of } b + 1 \)
Scaling up Edge Deletion

- Observation: Subgraphs induced from live-edge graphs are all TREES!
- $\text{Score}(edge) = |S| + \text{Supp} + \text{Support} + 1$

Stage 1: Top-Down BFS

Stage 2: Bottom-up Traversal

$O(V^2)$

$O(V)$
Scalability

Running Time (seconds per edge, $\log_2$)

Number of Nodes (log$_2$)

Edge Deletion
Edge Addition

Linear up to logarithmic factors

2 million edges

32 million edges

KDD, NYC August 26th 2014 Elias Khalil
More Experiments: Deletion

LOWER IS BETTER
More Experiments: 
Addition 

HIGHER IS BETTER
Conclusions

• We present:
  1. Network **optimization** problems under the LT model
  2. A better **theoretical** understanding of the LT model
  3. **Scalable**, near-optimal algorithms for the problems
  4. **State-of-the-art** experimental performance

Thank you!

lyes@gatech.edu