Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees

Alan Kuhnle\textsuperscript{1,*}, Victoria G. Crawford\textsuperscript{1}, My T. Thai\textsuperscript{1}

\textsuperscript{1}Computer and Information Science and Engineering, University of Florida, Gainesville, FL

\textsuperscript{*} kuhnle@ufl.edu

ACM SIGMETRICS 2018
LB MULTICUT: Given a network and a set of pairs of vertices, identify the minimum-size set of vertices (edges) whose removal will sufficiently separate each pair.

---

**LB MULTICUT**: Given a network and a set of pairs of vertices, identify the minimum-size set of vertices (edges) whose removal will sufficiently separate each pair.

“sufficiently separate” $\rightarrow d(s, t) > T$

---

**Introduction**

- 
  - **LB MULTICUT**: Given a network and a set of pairs of vertices, identify the minimum-size set of vertices (edges) whose removal will sufficiently separate each pair.
  - “sufficiently separate” → $d(s, t) > T$
  - **Motivation**: Network robustness.

---

LB MULTICUT: Given a network and a set of pairs of vertices, identify the minimum-size set of vertices (edges) whose removal will sufficiently separate each pair

“sufficiently separate” $\rightarrow d(s, t) > T$

Motivation: Network robustness

Distance as proxy for network functionality

---

Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees

Introduction

LB MULTICUT: Given a network and a set of pairs of vertices, identify the minimum-size set of vertices (edges) whose removal will sufficiently separate each pair

“sufficiently separate” → $d(s, t) > T$

Motivation: Network robustness

Distance as proxy for network functionality

For example, distance could encode packet loss in a communication network or travel time in a road network

---

**LB MULTICUT:** Given a network and a set of pairs of vertices, identify the minimum-size set of vertices (edges) whose removal will sufficiently separate each pair.

- “sufficiently separate” $\rightarrow d(s, t) > T$

**Motivation:** Network robustness

**Distance as proxy for network functionality**

For example, distance could encode packet loss in a communication network or travel time in a road network.

- $T$ can be input (LB MULTICUT) or fixed ($T$-MULTICUT)

---

Vulnerability assessment: LB MULTICUT

Figure: $S = \{(0, 12)\}, \ T = 5.$
Vulnerability assessment: LB MULTICUT

Figure: Want: min-size set of edges to remove s.t. $d(0, 12) > 5$
Vulnerability assessment: LB MULTICUT

**Figure:** Optimal solution has 2 edges.
Vulnerability assessment: LB MULTICUT

Figure: Classical cut of \((s, t)\) must take three edges
Vulnerability assessment: pseudo-separation

For multicut, vertex and edge versions are not equivalent.

**Figure**: Approximation results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Undirected</th>
<th>Directed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUT (both)</td>
<td>$1^2$</td>
<td>$1$</td>
</tr>
<tr>
<td>MULTICUT (edge)</td>
<td>$O(\log k)^3$</td>
<td>$O(n^{11/23})^4$</td>
</tr>
<tr>
<td>P. MULTICUT (edge)</td>
<td>$O(\log^2 n \log \log n)^5$</td>
<td>-</td>
</tr>
<tr>
<td>MULTICUT (vertex)</td>
<td>$O(\log k)^6$</td>
<td>-</td>
</tr>
<tr>
<td>T-CUT (both)</td>
<td>$T/2^7$</td>
<td>$T/2^7$</td>
</tr>
<tr>
<td>T-MULTICUT (both)</td>
<td>$T + 1^8$</td>
<td>$T + 1^8$</td>
</tr>
<tr>
<td>P. T-MULTICUT (both)</td>
<td>$(T + 1)/\epsilon^8$</td>
<td>$(T + 1)/\epsilon^8$</td>
</tr>
</tbody>
</table>

2. Elias, Feinstein, and Shannon; Ford, Fulkerson (1956)
Results of Kuhnle et al. (2017):
Results of Kuhnle et al. (2017):
- Formulated vulnerability assessment based upon LB MULTICUT
Results of Kuhnle et al. (2017):
- Formulated vulnerability assessment based upon LB MULTICUT
- Provided approximation algorithms when $T$ is fixed parameter
Results of Kuhnle et al. (2017):
- Formulated vulnerability assessment based upon LB MULTICUT
- Provided approximation algorithms when $T$ is fixed parameter
  - Require enumeration of all paths of length at most $T$
• Results of Kuhnle et al. (2017):
  • Formulated vulnerability assessment based upon LB MULTICUT
  • Provided approximation algorithms when $T$ is fixed parameter
    • Require enumeration of all paths of length at most $T$
  • Approximability lower bound of $2 - \epsilon$ (assuming UGC)
Contributions

- Primal-dual algorithm
Contributions

- Primal-dual algorithm
  - Approximation ratio: $T$, even when $T$ is input
Contributions

- Primal-dual algorithm
  - Approximation ratio: $T$, even when $T$ is input
  - Time complexity:
    \[ O\left(\frac{k}{D}\left(m + n\log(n)\right)\right) \]
    
    - Number of pairs
    - Max. degree
    - Dijkstra's alg.
Contributions

- Primal-dual algorithm
  - Approximation ratio: $T$, even when $T$ is input
  - Time complexity:
    \[
    O\left(\frac{k}{D} \cdot (m + n \log(n))\right)
    \]
    Number of pairs. Max. degree Dijkstra’s alg.

- Inapproximability result:
  \[
  \Omega(T),
  \]
  unless $NP \subseteq BPP$. 
Contributions

- Primal-dual algorithm
  - Approximation ratio: $T$, even when $T$ is input
  - Time complexity:

\[
O\left(\frac{k}{D}, \frac{D}{(m + n \log(n))}\right)
\]

  - Number of pairs. Max. degree. Dijkstra’s alg.

- Inapproximability result:

\[\Omega(T),\]

unless $NP \subseteq BPP$.

- Additional approximation algorithms.
Introduction

Primal-dual approach

- Primal-dual
Primal-dual approach

- **Primal-dual**
  - Until feasible,
Primal-dual approach

- **Primal-dual**
  - Until feasible,
  - remove (i.e. add to solution) an overall shortest path between any pair in $S$
Primal-dual approach

**Primal-dual**
- Until feasible,
- remove (i.e. add to solution) an overall shortest path between any pair in $S$
- To improve solution quality in practice, can prune it
Primal-dual approach

**Primal-dual**
- Until feasible,
  - remove (i.e. add to solution) an overall shortest path between any pair in $S$
- To improve solution quality in practice, can prune it
- “fully dynamic”: solution can be updated upon changes in edge weight and vertex insertion / deletion
Primal-dual, example

Figure: $d(0, 12) = 4$, $T = 5$
Primal-dual, example

Figure: Compute shortest path and add all edges into solution.
Primal-dual, example

Figure: Compute shortest path and add all edges into solution.
Primal-dual, example

Figure: $d(0, 12) = 6$, $T = 5$
Primal-dual, example

Figure: So we’re done.
Primal-dual, example

Figure: Can prune edges (5, 6), (6, 7)
Primal-dual, example

Figure: How to update?
Primal-dual, example

Figure: Add (2, 8)
Primal-dual, example

Figure: Add (2, 8)
Introduction

Primal-dual, example

Figure: $d(0, 12) = 5$, $T = 5$
Primal-dual, example

Figure: Add shortest path
Primal-dual, example

Figure: Prune
TAG Algorithm

Strategy:
- Maintains a maximal, edge-disjoint collection $U$ of paths
Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees

Introduction

TAG Algorithm

Strategy:
- Maintains a maximal, edge-disjoint collection $U$ of paths
- Number of paths lower bounds $OPT$
TAG Algorithm

Strategy:
- Maintains a maximal, edge-disjoint collection $U$ of paths
- Number of paths lower bounds $\text{OPT}$
- Ratio $T$
Theorem

Let $T \geq 16$. Unless $\text{NP} \subseteq \text{BPP}$, there is no polynomial-time algorithm to approximate $T$-LB MULTICUT within a factor of

$$\left\lfloor \frac{T}{6} \right\rfloor - 1 - \epsilon,$$

for any $\epsilon > 0$.

- Approximation-preserving reduction from Cycle Interdiction Problem: minimum set of edges to intersect all cycles of length at most $r$. Guruswami et al. showed lower bound of

$$\left\lfloor \frac{r}{2} \right\rfloor - 1 - \epsilon.$$
Theorem

Let $T \geq 16$. Unless $NP \subseteq BPP$, there is no polynomial-time algorithm to approximate $T$–LB MULTICUT within a factor of

$$\left\lfloor \frac{T}{6} \right\rfloor - 1 - \epsilon,$$

for any $\epsilon > 0$.

- Approximation-preserving reduction from Cycle Interdiction Problem: minimum set of edges to intersect all cycles of length at most $r$. Guruswami et al. showed lower bound of

$$\left\lfloor \frac{r}{2} \right\rfloor - 1 - \epsilon.$$

- Requires $\Omega(n)$ pairs.
**Theorem**

Let $T \geq 16$. Unless $NP \subseteq BPP$, there is no polynomial-time algorithm to approximate $T$–LB MULTICUT within a factor of

$$\left\lfloor \frac{T}{6} \right\rfloor - 1 - \epsilon,$$

for any $\epsilon > 0$.

- Approximation-preserving reduction from Cycle Interdiction Problem: minimum set of edges to intersect all cycles of length at most $r$. Guruswami et al. showed lower bound of

$$\left\lfloor \frac{r}{2} \right\rfloor - 1 - \epsilon.$$

- Requires $\Omega(n)$ pairs.
- For single pair $T$-CUT, best lower bound is currently $\sqrt{T}$. 
Experimental Evaluation

Figure: Static experiments on ER 100 networks.
## Experimental Evaluation

### Table: Real-world traces.

| Network   | $|V|$         | $|E|$         | $\hat{d}(x, y)$ | Weighted |
|-----------|--------------|--------------|----------------|----------|
| Gnutella  | $6.301 \times 10^3$ | $2.078 \times 10^4$ | 4.63           | No       |
| Enron     | $3.669 \times 10^4$ | $1.838 \times 10^5$ | 4.04           | No       |
| RoadSF    | $1.748 \times 10^5$ | $2.218 \times 10^5$ | 3635.29        | Yes      |
| Google    | $8.757 \times 10^5$ | $4.322 \times 10^6$ | 6.33           | No       |
| Skitter   | $1.696 \times 10^6$ | $1.109 \times 10^7$ | 13.00          | Yes      |
| Friendster| $1.248 \times 10^8$ | $1.806 \times 10^9$ | 4.99           | No       |
Figure: RoadSF: (a) Solution Size, (b) Running time vs. number of pairs $k$
### Table: Static average add/remove results (TAG).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Addition time (s)</th>
<th>Removal time (s)</th>
<th>Addition size</th>
<th>Removal size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gnutella</td>
<td>0.6</td>
<td>0.2</td>
<td>125.4</td>
<td>30.0</td>
</tr>
<tr>
<td>Enron</td>
<td>21.6</td>
<td>3.9</td>
<td>269.8</td>
<td>76.2</td>
</tr>
<tr>
<td>Google</td>
<td>3765.9</td>
<td>696.4</td>
<td>617.0</td>
<td>155.4</td>
</tr>
<tr>
<td>Friendster</td>
<td>38127.0</td>
<td>32465.0</td>
<td>94.0</td>
<td>49.6</td>
</tr>
</tbody>
</table>

### Table: Dynamic average add/remove results (TAG).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Addition time (ms)</th>
<th>Removal time (ms)</th>
<th>Addition loss</th>
<th>Removal loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gnutella</td>
<td>1.0</td>
<td>1.0</td>
<td>6.7%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Enron</td>
<td>3.5</td>
<td>0.7</td>
<td>7.0%</td>
<td>15.5%</td>
</tr>
<tr>
<td>Google</td>
<td>110.0</td>
<td>12.0</td>
<td>20.3%</td>
<td>13.3%</td>
</tr>
<tr>
<td>Friendster</td>
<td>1100.0</td>
<td>0.2</td>
<td>0.0%</td>
<td>2.8%</td>
</tr>
</tbody>
</table>
Summary

- Scalable and dynamic algorithm with ratio $T$
Summary

- Scalable and dynamic algorithm with ratio $T$
- Lower bound of $T/6 - 1$, $T \geq 16$
Summary

- Scalable and dynamic algorithm with ratio $T$
- Lower bound of $T/6 - 1$, $T \geq 16$
- Implementations available:
  https://gitlab.com/kuhnle/multi-pcut
Summary

- Scalable and dynamic algorithm with ratio $T$
- Lower bound of $T/6 - 1$, $T \geq 16$
- Implementations available: https://gitlab.com/kuhnle/multi-pcut
- Future work: improve lower bound or better approximation?
Thank you! Questions?

Alan Kuhnle, kuhnle@ufl.edu
References I

An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms.  

Improved approximation for directed cut problems.  

A unified approach to approximating partial covering problems.  

Multiway Cuts in Directed and Node Weighted Graphs.  

Length-Bounded Cuts and Flows.  

Pseudo-separation for assessment of structural vulnerability of a network.  