Global and fixed-terminal cuts in digraphs

Kristóf Bérczi, Karthekeyan Chandrasekaran, Tamás Király, Euiwoong Lee and Chao Xu
August 2, 2017

University of Illinois, Urbana-Champaign. Department of Computer Science.
What to expect

• Survey talk.
• Lot of definitions, problems and examples.
• Few technical details.
• Many open problems.
Let $G = (V, E)$ be a graph.

A cut (node cut) is a set of edges (nodes) that disconnects some pair of vertices if removed. The value of a cut (node cut) is the number of edges (nodes) in the cut.

**min global cut (node cut) problem:**

- Input: $G$.
- Output: Minimum (node) cut.

**min local cut (node cut) problem:**

- Input: $G, s, t \in V$.
- Output: Minimum (node) cut that disconnects $s$ and $t$. 
Complexity separation of local and global cuts

- local cut at least as hard as global cut.
- min cut can be reduced to $O(n^2)$ calls to min local cut.
- Main problem: When is local cut strictly harder than global cut?
  - Local cut is NP-hard but polynomial time algorithm for global cut.
  - $\alpha$-inapproximability of local cut, but $\alpha - \delta$-approximation for global cut for $\delta > 0$. 
Local and global cuts in undirected graphs
Consider a undirected graph $G = (V, E)$. We say $s, t \in V$ are disconnected if there is no paths between $s$ and $t$.

- **cut**: A set of edges s.t. its removal disconnects some pair of vertices.
- **st-cut**: A set of edges s.t. its removal disconnects $s$ and $t$.
- Each set $S \subset V \setminus \{t\}$ such that $s \in S$ determines a st-cut with $|\delta(S)|$ edges.
- $\lambda(s, t; G)$ is the value of min st-cut.
- $\lambda(G) = \min_{s, t \in V} \lambda(s, t; G)$ is the value of min cut.
Algorithmic aspects

- $\lambda(s, t) = \min_{s \in S} |\delta(S)|$ can be computed in $O(nm)$ time by reducing to maximum flow. [Orlin 2013]
- $\lambda(G)$ can be computed directly using MA-ordering. [Nagamochi & Ibaraki 1992]
Node cuts

- $\kappa(s, t; G)$ is the value of min $st$-node-cut.
- $\kappa(G) = \min_{s, t \in V} \kappa(s, t; G)$ is the value of min node-cut.

Both can be solved in polynomial time, since $\kappa(s, t; G)$ reduces to maximum flow.
Both global and local problem for edge and node deletion can be solved in polynomial time.
Local and global cuts in digraphs
Two definitions of $st$-cut

For a graph $G$ and two vertices $s$ and $t$. A $st$-cut is a set of edges $E'$, s.t. in $G - E'$

- Definition 1: There is no path between $s$ and $t$.
- Definition 2: There is no vertex that can reach both $s$ and $t$.

Two notions are the same in undirected graphs.
Consider a digraph $G = (V, E)$.

**Definition (st-bicut)**
A set of edges $E'$ is a st-bicut, if there is no path between $s$ and $t$ in $G - E'$

**Definition (st-double cut)**
A set of edges $E'$ is a st-double cut, if there is no vertex $v \in V$ that can reach both $s$ and $t$. 
Example
$st$-bicut example
Not a $st$-double cut
st-double cut example
- disconnected $s$ and $t$: No vertex can reach both $s$ and $t$.
- $\lambda_d(s, t; G)$ is the value of min $st$-double cut.
- $\lambda_d(G) = \min_{s,t \in V} \lambda_d(s, t; G)$. 

$st$-Double cut
Let $E'$ be a st-double cut in $G$.

- $E'$ is a st-bicut.
- $G - E'$ has no arborescence.

$\lambda_d(G)$ is the minimum number of edges to remove to destroy all arborescence.
Why double cut?

- Blocking arborescence [Bernáth & Pap 2013]
- Application in distributed computing.

**Theorem ([Tseng & Vaidya 2015])**

*The consensus problem in synchronized model can tolerate* $f$ *edge (node) failure iff remove any* $f$ *edges (nodes), there is still an arborescence.*

- The largest edge failure tolerance is $\lambda_d(G) - 1$.
- The largest node failure tolerance is $\kappa_d(G) - 1$. 
How to think about double cut

Theorem ([Bernáth & Pap 2013])
Finding $\lambda_d(s, t)$ is equivalent to finding disjoint sets $S, T \subseteq V$, such that $s \in S$, $t \in T$ and $d^{in}(S) + d^{in}(T)$ is minimized.

Proof.

• $\Rightarrow$ Let set of vertices that can reach $s$ and $t$ to be $S$ and $T$, respectively.

• $\Leftarrow$ Remove incoming edges to $S$ and $T$ then no vertices outside $S$ can reach $s$, outside $t$ can reach $T$.

Corollary
$\lambda_d(s, t)$ can be computed in polynomial time by reducing to maximum flow.
Finding $\lambda_d(s, t)$ through max-flow
There is no complexity separation for local and global double cut.
Node double cut

- $\kappa_d(s, t; G)$ is the value of min st-node double cut.
- $\kappa_d(G) = \min_{s, t \in V} \kappa_d(s, t; G)$.

Difficulty: non-monotonic. $A$ is a node-double cut, $A \cup \{v\}$ might not be a node-double cut.

$A = \emptyset$. 
Node Double cut results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Inapproximability</th>
</tr>
</thead>
<tbody>
<tr>
<td>node double cut</td>
<td>2</td>
<td>$3/2 - \epsilon$</td>
</tr>
<tr>
<td>node st-double cut</td>
<td>2</td>
<td>$2 - \epsilon$</td>
</tr>
</tbody>
</table>

Open: Is node double cut strictly harder than node st-double cut?
Bicut
Bicut

*$st$*-bicut: a set of edges such that after its removal there is no path between $s$ and $t$.

- $\lambda_b(s, t)$ for the value of $\min st$-bicut.
- $\lambda_b(G) = \min_{s,t \in V} \lambda_b(s, t)$
A special case of multicut in directed graphs.

- 2-approximation possible. [Dahlhaus et. al. 1994]
- $2 - \epsilon$-inapproximable under UGC. [Chekuri & Madan 16, Lee 16]
How to think about global bicut

A and B are uncomparable if $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$.

Theorem
The min-bicut problem is equivalent to finding a uncomparable pair $A, B \subseteq V$ with minimum $|\delta^{in}(A) \cup \delta^{in}(B)|$.

Proof.

• $\Rightarrow$ Remove $\delta^{in}(A) \cup \delta^{in}(B)$, then nodes in $A \setminus B$ cannot reach nodes in $B \setminus A$ and vice versa.

• $\Leftarrow$ no path between $s$ and $t$. The set of nodes that can reach $s$ and the set of nodes that can reach $t$ are uncomparable, and have in-degree $0$. 

\[\square\]
Theorem ([BCKLX 2017])

A $(2 - \delta)$-approximation, where $\delta \geq \frac{1}{448}$.

A separation between local and global bicut! It’s not known if computing $\lambda_b(G)$ is NP-hard.
A cut is a \textbf{s*-bicut} if it is a \textit{st}-bicut for some \( t \).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Inapproximability</th>
</tr>
</thead>
<tbody>
<tr>
<td>bicut</td>
<td>( 2 - \delta )</td>
<td>( ? )</td>
</tr>
<tr>
<td>\textbf{s*-bicut}</td>
<td>2</td>
<td>4/3 - ( \epsilon )</td>
</tr>
<tr>
<td>\textit{st}-bicut</td>
<td>2</td>
<td>2 - ( \epsilon )</td>
</tr>
</tbody>
</table>
st-node-bicut: a set of nodes such that after its removal there is no path between $s$ and $t$.

- $\kappa_b(s, t)$ for the value of min st-bicut.
- $\kappa_b(G) = \min_{s,t\in V} \kappa_b(s, t)$. 
min $st$-node bicut and min $st$-bicut are equivalent

- Reduce $st$-bicut to $st$-node bicut: Split each edge by a node, original node have infinite weight.
- Reduce $st$-node bicut to $st$-bicut: Split each node and add edge weight equal to the node weight. All other edges have infinite weight.

The equivalence doesn’t hold for global node bicut and global bicut.
### Node Bicut results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Inapproximability</th>
</tr>
</thead>
<tbody>
<tr>
<td>node bicut</td>
<td>2</td>
<td>$3/2 - \epsilon$</td>
</tr>
<tr>
<td>node s*-bicut</td>
<td>2</td>
<td>$3/2 - \epsilon$</td>
</tr>
<tr>
<td>node st-bicut</td>
<td>2</td>
<td>$2 - \epsilon$</td>
</tr>
</tbody>
</table>

Open: Is there a complexity separation between node bicut and st-node bicut?
Disconnecting more than 2 vertices
A set $T \subset V$ is disconnected if there is no path between any pair of vertices in $T$.

- $k$-cut: A set of edges s.t. its removal creates a disconnected set of size at least $k$.
- $T$-separating $k$-cut: A $k$-cut s.t. its removal disconnects $T$.
- $\lambda^k(T; G)$ is the value of min $T$-separating $k$-cut.
- Local: $\lambda^k(T; G)$ for $|T| = k$.
- Global: $\lambda^k(G)$, is the value of min $k$-cut.

$$\lambda^k(G) = \min_{|T|=k, T \subset V} \lambda^k(T; G).$$

$\lambda^k(G) = \lambda^k(T, G)$ where $|T| \leq 1$.

- Semiglobal: $|T| = i$ for some $2 \leq i < k$. 
Let $k$ be a constant.

- Computing $\lambda^k(G)$ can be done in polynomial time. [Goldschmidt & Hochbaum 1994]

- Computing $\lambda^k(T; G)$ is hard for $|T| \geq 3$, but admits a 2-approximation. [Garg, Vazirani & Yannakakis 2004] In particular, min local $k$-cut for $k \geq 3$ is NP-hard.

Local $k$-cut is strictly harder than the global $k$-cut.
A subproblem for bicut approximation: $s \ast t$-linear 3-cut

A set of edges $E'$ is $s \ast t$-linear 3-cut if there exist a vertex $r \neq s, t$, such that $s$ cannot reach $r$ and $t$, and $r$ cannot reach $t$ in $G - E'$.

**Theorem**

*Finding $s \ast t$-linear 3-cut is equivalent to finding*

$$\min \left\{ d(A, B) + d(A, C) + d(B, C) : \{A, B, C\} \text{ a partition of } V, \ A, B, C \neq \emptyset, s \in A, t \in C \right\}.$$
Why do we care about $s \ast t$ linear 3-cut

- It’s a semiglobal version of linear $k$-cut. [Chekuri & Madan 2017]
- Improvement in approximation of $s \ast t$ linear 3-cut improves the $2 - \delta$ approximation algorithm for min bicut.
• Finding a minimum $s \ast t$-linear 3-cut is not known to be NP-hard.
• We’ve shown there is a $3/2$-approximation algorithm.
• A newer result shows there is a $\sqrt{2}$-approximation algorithm. [Bérczi et. al. unpublished]

Is $s \ast t$-linear 3-cut NP-hard?

What about the undirected version?
What is the undirected version of $s \star t$-linear 3-cut?
What is the undirected version of $s \ast t$-linear 3-cut?

Undirected version of $s \ast t$-linear 3-cut is \{s, t\}-separating 3-cut.
\{s, t\}-separating edge $k$-cut

We want to find $\lambda^k(T; G)$ for $|T| = 2$.

1. At the boundary between P and NP-hard.
   • Finding $\lambda^k(T, G)$ is NP-hard for $|T| \geq 3$.
   • Finding $\lambda^k(T, G)$ is easy for $|T| \leq 1$.

2. An open problem [Queyranne 2012].
**Theorem ([BCKLX 2017])**

Let \( \{V_1, \ldots, V_k\} \) be a partition of \( V \) corresponding to an optimal solution of \( \text{min } \text{st-separating } k \)-cut in \( G \). \( s \in V_{k-1} \) and \( t \in V_k \). Add a infinite weight edge between \( st \) and call the new graph \( H \).

\[
c(V_1, \ldots, V_{k-2}, V_{k-1} \cup V_k) \leq 2\lambda_{k-1}(H)
\]
Proof

Let \( W_1, \ldots, W_{k-1} \) be an optimal \( k-1 \) cut for \( H \) and \( s, t \in W_{k-1} \). Let \( U_1 \) and \( U_2 \) be min \( st \)-cut in \( G[W_{k-1}] \)
\[
\lambda^k(G) \leq c(W_1, \ldots, W_{k-2}, U_1, U_2) = \lambda^{k-1}(H) + \lambda(s, t; G[W_{k-1}]).
\]
Proof

Let $V_1, \ldots, V_k$ be an optimal $k$-cut for $G$, $s \in V_{k-1}$, $t \in V_k$.

$$
\lambda(s, t; G) \leq d(V_{k-1}, V_k) + \frac{1}{2} \left( d(V_1 \cup \ldots \cup V_{k-2}) + \sum_{i,j \leq k-2, i \neq j} d(V_i, V_j) \right)
$$
Proof

\[
\lambda^{k-1}(H) + \lambda(s, t; G[W_{k-1}]) \\
\geq \lambda^k(G) \\
= d(V_{k-1}, V_k) + d(V_1 \cup \ldots \cup V_{k-2}) + \sum_{i,j \leq k-2, i \neq j} d(V_i, V_j) \\
\geq \lambda(s, t; G) + \frac{1}{2} \left( d(V_1 \cup \ldots \cup V_{k-2}) + \sum_{i,j \leq k-2, i \neq j} d(V_i, V_j) \right) \\
= \lambda(s, t; G) + \frac{1}{2} \left( c(V_1, \ldots, V_{k-2}, V_{k-1} \cup V_k) \right) \\
\geq \lambda(s, t; G[W_{k-1}]) + \frac{1}{2} \left( c(V_1, \ldots, V_{k-2}, V_{k-1} \cup V_k) \right)
\]
Algorithm for \textit{st}-separating \textit{k}-cut

Algorithm

1. Enumerate all \(k - 1\)-cut \(\{W_1, \ldots, W_{k-1}\}\) with value at most \(2\lambda^{k-1}(H)\), assuming \(s, t \in W_{k-1}\).
2. For each \(k - 1\)-cut, find min-st-cut in \(G[W_{k-1}]\), say \(\{U_1, U_2\}\). Let \(\{W_1, \ldots, W_{k-2}, U_1, U_2\}\) be a candidate solution.
3. Output the candidate solution with the smallest value.

There are \(O(n^{2(k-1)})\) \(k - 1\)-partitions with value \(\leq 2\lambda^{k-1}(H)\).

[Karger & Stein 1996]

\textbf{Theorem ([BCKLX 2017])}

The \textit{st}-separating \textit{k}-cut can be solved in polynomial time for constant \textit{k}.
Node $k$-cuts

1. $\kappa^k(T; G)$ is the value of the minimum $T$-separating node $k$-cut.

2. $\kappa^k(G)$ is the value of the minimum node $k$-cut.
1. $\kappa^k(T; G)$ has a $(2 - 2/k)$-approximation [Garg, Vazirani & Yannakakis 2004].

2. It was raised as an open problem if $\kappa^k(G)$ is solvable in polynomial time for all $k \geq 3$. [Goldschmidt & Hochbaum 1994]
A complete characterization for node-$k$-cut.

**Theorem ([BCKLX 2017])**

If $k \geq 3$, then there exist a $(2 - 2/k)$-approximation algorithm for $\kappa^k(T; G)$ and cannot be approximated within $(2 - 2/k - \epsilon)$. Otherwise, it’s polynomial time solvable.
$(H, t)$-cuts
$(H, t)$-cuts

- $H$ a graph (digraph) on $\{1, \ldots, k\}$, and a integer $t$. $H$ is called the pattern graph.
- $G = (V, E)$ be a input graph (digraph)
- A $k$-partition $(V_1, \ldots, V_k)$ of $V$ where $V_{t+1}, \ldots, V_k$ are non-empty is a $(H, t)$-cut. ($V_1, \ldots, V_t$ can be empty)
- The $(H, t)$-cut value of $(V_1, \ldots, V_k)$ is

$$
\sum_{e \in V_i \rightarrow V_j} w(e)
\quad (i, j) \in E(H)
$$

- What can we model with $(H, t)$-cut?
\( k \)-cut

\[ t = 0 \]
Find $A$ and $B$ such that $A \cap B = \emptyset$ and $|\delta^{in}(A)| + |\delta^{in}(B)|$ is minimized.
Linear 3-cut

\[ t = 0. \]
$t = 2$. Find two incomparable sets $A$ and $B$ such that $|\delta^{in}(A) \cup \delta^{in}(B)|$ is minimized. Let $V_1 = V \setminus (A \cup B)$, $V_2 = A \cap B$, $V_3 = A \setminus B$, $V_4 = B \setminus A$. 
$k$-subpartition

Find $k$ sets $\{V_1, \ldots, V_k\}$ such that $V_i \cap V_j = \emptyset$ and minimize

$$\sum_{i=1}^{k} |\delta^{in}(V_i)|.$$  

Double cut is equivalent to 2-subpartition.

$t = 1$. Solvable in polynomial time if $G$ is obtained from bidirect all edges of a undirected graph [Nagamochi 2007].
Polynomial time solvable cases when $H$ is undirected

- If $H$ has at most 4 vertices, then finding $\text{min-}(H, 0)$-cut is NP-hard iff $H = 2K_2$. [Elem, Hassin & Monnot 2013 unpublished]
  Reduces to partition the graph to two disjoint bicliques.

A vertex $v$ is **neighborhood minimal**, if there is no vertex $u$ such that $N(u) \subset N(v)$. $\text{min-}(H, 0)$-cut is solvable in polynomial time if

- The neighborhood minimal vertices of $H$ is a independent set in $H$.
- $H = H_1 + H_2$ where $\text{min-}(H_1, 0)$-cut and $\text{min-}(H_2, 0)$-cut are solvable in polynomial time. [Kawarabayashi and X unpublished]
Fixing terminals

Given $G$ and $U_1, \ldots, U_k$, find min-$(H, t)$-cut $(V_1, \ldots, V_k)$ such that $U_i \subseteq V_i$. 
Open Problems
Polynomial time algorithms for $(H, t)$-cut

- For which $(H, t)$ pair is min $(H, t)$-cut solvable in polynomial time?
- Does $(H, 0)$-cut solvable in polynomial time implies $(H, t)$-cut solvable in polynomial time for all $t$?
- What about fixed terminal version?
### Close the gaps

<table>
<thead>
<tr>
<th>Problem</th>
<th>Edge-deletion</th>
<th>Node-deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DOUBLECut</strong></td>
<td>Poly-time</td>
<td>2-approx (3/2 − ε)-inapprox</td>
</tr>
<tr>
<td><strong>st-DOUBLECut</strong></td>
<td>Poly-time</td>
<td>2-approx (2 − ε)-inapprox</td>
</tr>
<tr>
<td><strong>BiCut</strong></td>
<td>(2 − 1/448)-approx NP-hard?</td>
<td>2-approx (3/2 − ε)-inapprox</td>
</tr>
<tr>
<td><em><em>s</em>-BiCut</em>*</td>
<td>2-approx (4/3 − ε)-inapprox</td>
<td>2-approx (3/2 − ε)-inapprox</td>
</tr>
<tr>
<td><strong>st-BiCut</strong></td>
<td>2-approx (2 − ε)-inapprox</td>
<td>Equivalent to edge-deletion</td>
</tr>
<tr>
<td><strong>S * t-LINEAR 3-CUT</strong></td>
<td>√2-approx NP-hard?</td>
<td>2-approx (4/3 − ε)-inapprox</td>
</tr>
</tbody>
</table>
$\lambda^k(G)$ can be found in hypergraphs in randomized polynomial time [Chandrasekaran, X & Yu unpublished].

What about $\lambda^k(\{s, t\} ; G)$?

- The algorithm is still correct.
- Number of approximate min-$k$-cut is exponential.
- Exponential running time.

Can we find $\lambda^k(\{s, t\} ; G)$ in polynomial time for hypergraphs?
Thank You!