Important cuts

Dániel Marx

Lecture #10
January 11, 2020
Overview

Main message
Small cuts in graphs have interesting extremal properties that can be exploited in combinatorial and algorithmic results.

- Bounding the number of “important” cuts.
- Edge/vertex versions, directed/undirected versions, undeletable edges/vertices
- “directed edge” or “arc”
- Algorithmic applications: FPT algorithm for
  - Multiway cut
  - Directed Feedback Vertex Set
Definition: \( \delta(R) \) is the set of edges with exactly one endpoint in \( R \).

Definition: A set \( S \) of edges is a \textbf{minimal \( (X, Y) \)-cut} if there is no \( X - Y \) path in \( G \setminus S \) and no proper subset of \( S \) breaks every \( X - Y \) path.

Observation: Every minimal \( (X, Y) \)-cut \( S \) can be expressed as \( S = \delta(R) \) for some \( X \subseteq R \) and \( R \cap Y = \emptyset \).
Theorem

A minimum \((X, Y)\)-cut can be found in polynomial time.

Theorem

The size of a minimum \((X, Y)\)-cut equals the maximum size of a pairwise edge-disjoint collection of \(X - Y\) paths.
Finding minimum cuts

There is a long list of algorithms for finding disjoint paths and minimum cuts.

- Edmonds-Karp: $O(|V(G)| \cdot |E(G)|^2)$
- Dinitz: $O(|V(G)|^2 \cdot |E(G)|)$
- Push-relabel: $O(|V(G)|^3)$
- Orlin-King-Rao-Tarjan: $O(|V(G)| \cdot |E(G)|)$
- ...
- Liu-Sidford: $O(|E(G)|^{4/3} U^{1/3})$

But we need only the following result:

**Theorem**

An $(X, Y)$-cut of size at most $k$ (if exists) can be found in time $O(k \cdot (|V(G)| + |E(G)|))$. 
Finding minimum cuts

**Theorem**

An \((X, Y)\)-cut of size at most \(k\) (if exists) can be found in time \(O(k \cdot (|V(G)| + |E(G)|))\).

We try to grow a collection \(\mathcal{P}\) of edge-disjoint \(X - Y\) paths.

**Residual graph:**
- not used by \(\mathcal{P}\): bidirected,
- used by \(\mathcal{P}\): directed in the opposite direction.

![Diagram](image)
Finding minimum cuts

**Theorem**

An \((X, Y)\)-cut of size at most \(k\) (if exists) can be found in time \(O(k \cdot (|V(G)| + |E(G)|))\).

We try to grow a collection \(\mathcal{P}\) of edge-disjoint \(X - Y\) paths.

**Residual graph:**

- not used by \(\mathcal{P}\): bidirected,
- used by \(\mathcal{P}\): directed in the opposite direction.

![Diagram of original and residual graph with nodes and edges highlighted]
Finding minimum cuts

Theorem

An \((X, Y)\)-cut of size at most \(k\) (if exists) can be found in time \(O(k \cdot (|V(G)| + |E(G)|))\).

We try to grow a collection \(\mathcal{P}\) of edge-disjoint \(X - Y\) paths.

**Residual graph:**
- not used by \(\mathcal{P}\): bidirected,
- used by \(\mathcal{P}\): directed in the opposite direction.
Finding minimum cuts

**Theorem**

An \((X, Y)\)-cut of size at most \(k\) (if exists) can be found in time \(O(k \cdot (|V(G)| + |E(G)|))\).

We try to grow a collection \(\mathcal{P}\) of edge-disjoint \(X - Y\) paths.

**Residual graph:**

- not used by \(\mathcal{P}\): bidirected,
- used by \(\mathcal{P}\): directed in the opposite direction.

If we cannot find an augmenting path, we can find a (minimum) cut of size \(|\mathcal{P}|\).
Submodularity

**Fact:** The function $\delta$ is submodular: for arbitrary sets $A, B$,

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$$
Submodularity

**Fact:** The function $\delta$ is **submodular:** for arbitrary sets $A, B$,

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$$

**Proof:** Determine separately the contribution of the different types of edges.
Submodularity

**Fact:** The function $\delta$ is submodular: for arbitrary sets $A, B,$

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$$

**Proof:** Determine separately the contribution of the different types of edges.
Submodularity

**Fact:** The function $\delta$ is **submodular**: for arbitrary sets $A, B$,

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$$

**Proof:** Determine separately the contribution of the different types of edges.
**Submodularity**

**Fact:** The function $\delta$ is **submodular**: for arbitrary sets $A, B$,

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)| \quad 0 \quad 1 \quad 0 \quad 1$$

**Proof:** Determine separately the contribution of the different types of edges.
Submodularity

Fact: The function $\delta$ is submodular: for arbitrary sets $A, B$,

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$$

Proof: Determine separately the contribution of the different types of edges.
Submodularity

**Fact:** The function $\delta$ is submodular: for arbitrary sets $A, B$,

$$\frac{|\delta(A)|}{1} + \frac{|\delta(B)|}{1} \geq \frac{|\delta(A \cap B)|}{1} + \frac{|\delta(A \cup B)|}{1}$$

**Proof:** Determine separately the contribution of the different types of edges.
Submodularity

Fact: The function $\delta$ is submodular: for arbitrary sets $A, B$,

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$$

Proof: Determine separately the contribution of the different types of edges.
Submodularity

Lemma

Let $\lambda$ be the minimum $(X, Y)$-cut size. There is a unique maximal $R_{\text{max}} \supseteq X$ such that $\delta(R_{\text{max}})$ is an $(X, Y)$-cut of size $\lambda$.
Submodularity

Lemma

Let $\lambda$ be the minimum $(X, Y)$-cut size. There is a unique maximal $R_{\text{max}} \supseteq X$ such that $\delta(R_{\text{max}})$ is an $(X, Y)$-cut of size $\lambda$.

Proof: Let $R_1, R_2 \supseteq X$ be two sets such that $\delta(R_1), \delta(R_2)$ are $(X, Y)$-cuts of size $\lambda$.

$$|\delta(R_1)| + |\delta(R_2)| \geq |\delta(R_1 \cap R_2)| + |\delta(R_1 \cup R_2)|$$

$$\lambda \quad \lambda \quad \geq \lambda$$

$$\Rightarrow |\delta(R_1 \cup R_2)| \leq \lambda$$

Note: Analogous result holds for a unique minimal $R_{\text{min}}$. 


Finding $R_{\text{min}}$ and $R_{\text{max}}$

**Lemma**

Given a graph $G$ and sets $X, Y \subseteq V(G)$, the sets $R_{\text{min}}$ and $R_{\text{max}}$ can be found in polynomial time.

**Proof:** Iteratively add vertices to $X$ if they do not increase the minimum $X - Y$ cut size. When the process stops, $X = R_{\text{max}}$. Similar for $R_{\text{min}}$.

But we can do better!
Finding $R_{\text{min}}$ and $R_{\text{max}}$

**Lemma**

Given a graph $G$ and sets $X, Y \subseteq V(G)$, the sets $R_{\text{min}}$ and $R_{\text{max}}$ can be found in $O(\lambda \cdot (|V(G)| + |E(G)|))$ time, where $\lambda$ is the minimum $X - Y$ cut size.

**Proof:** Look at the residual graph.

$R_{\text{min}}$: vertices reachable from $X$.

$R_{\text{max}}$: vertices from which $Y$ is not reachable.
Important cuts

**Definition:** $\delta(R)$ is the set of edges with exactly one endpoint in $R$.

**Definition:** A set $S$ of edges is a **minimal $(X, Y)$-cut** if there is no $X-Y$ path in $G \setminus S$ and no proper subset of $S$ breaks every $X-Y$ path.

**Observation:** Every minimal $(X, Y)$-cut $S$ can be expressed as $S = \delta(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$. 

![Diagram showing a network with important cuts and minimal cuts](image-url)
Important cuts

**Definition**

A minimal \((X, Y)\)-cut \(\delta(R)\) is **important** if there is no \((X, Y)\)-cut \(\delta(R')\) with \(R \subset R'\) and \(|\delta(R')| \leq |\delta(R)|\).

**Note**: Can be checked in polynomial time if a cut is important.
Important cuts

Definition

A minimal \((X, Y)\)-cut \(\delta(R)\) is important if there is no \((X, Y)\)-cut \(\delta(R')\) with \(R \subset R'\) and \(|\delta(R')| \leq |\delta(R)|\).

Note: Can be checked in polynomial time if a cut is important.
Important cuts

Definition
A minimal \((X, Y)\)-cut \(\delta(R)\) is important if there is no \((X, Y)\)-cut \(\delta(R')\) with \(R \subset R'\) and \(|\delta(R')| \leq |\delta(R)|\).

Note: Can be checked in polynomial time if a cut is important.
Important cuts

**Definition**

A minimal $(X, Y)$-cut $\delta(R)$ is **important** if there is no $(X, Y)$-cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \leq |\delta(R)|$.

**Note:** Can be checked in polynomial time if a cut is important.

**Observation:** There is a unique important $(X, Y)$-cut of minimum size: $\delta(R_{\text{max}})$. 
Important cuts

The number of important cuts can be exponentially large.

Example:

This graph has $2^{k/2}$ important $(X, Y)$-cuts of size at most $k$. 
Important cuts

The number of important cuts can be exponentially large.

Example:

This graph has $2^{k/2}$ important $(X, Y)$-cuts of size at most $k$.

Theorem

There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$. 
Important cuts

Theorem
There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

Proof: Let $\lambda$ be the minimum $(X, Y)$-cut size and let $\delta(R_{\text{max}})$ be the unique important cut of size $\lambda$ such that $R_{\text{max}}$ is maximal.

(1) We show that $R_{\text{max}} \subseteq R$ for every important cut $\delta(R)$. 

\[ \lambda \geq \lambda \quad \Rightarrow \quad |\delta(R_{\text{max}} \cup R)| \leq |\delta(R)| \]

If $R \neq R_{\text{max}} \cup R$, then $\delta(R)$ is not important.
There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

**Proof:** Let $\lambda$ be the minimum $(X, Y)$-cut size and let $\delta(R_{\text{max}})$ be the unique important cut of size $\lambda$ such that $R_{\text{max}}$ is maximal.

(1) We show that $R_{\text{max}} \subseteq R$ for every important cut $\delta(R)$.

By the submodularity of $\delta$:

$$|\delta(R_{\text{max}})| + |\delta(R)| \geq |\delta(R_{\text{max}} \cap R)| + |\delta(R_{\text{max}} \cup R)| \geq \lambda \downarrow$$

$$|\delta(R_{\text{max}} \cup R)| \leq |\delta(R)| \downarrow$$

If $R \neq R_{\text{max}} \cup R$, then $\delta(R)$ is not important.
Important cuts

**Theorem**

There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

**Proof:** Let $\lambda$ be the minimum $(X, Y)$-cut size and let $\delta(R_{\text{max}})$ be the unique important cut of size $\lambda$ such that $R_{\text{max}}$ is maximal.

(1) We show that $R_{\text{max}} \subseteq R$ for every important cut $\delta(R)$.

By the submodularity of $\delta$:

$$|\delta(R_{\text{max}})| + |\delta(R)| \geq |\delta(R_{\text{max}} \cap R)| + |\delta(R_{\text{max}} \cup R)| \geq \lambda$$

$\Downarrow$

$$|\delta(R_{\text{max}} \cup R)| \leq |\delta(R)|$$

$\Downarrow$

If $R \neq R_{\text{max}} \cup R$, then $\delta(R)$ is not important.

Thus the important $(X, Y)$- and $(R_{\text{max}}, Y)$-cuts are the same.

$\Rightarrow$ We can assume $X = R_{\text{max}}$.  

12
(2) Search tree algorithm for enumerating all these cuts:
An (arbitrary) edge $uv$ leaving $X = R_{\text{max}}$ is either in the cut or not.

Branch 1: If $uv \in S$, then $S \setminus uv$ is an important $(X, Y)$-cut of size at most $k - 1$ in $G \setminus uv$.

$\Rightarrow k$ decreases by one, $\lambda$ decreases by at most 1.

Branch 2: If $uv \not\in S$, then $S$ is an important $(X \cup v, Y)$-cut of size at most $k$ in $G$.

$\Rightarrow k$ remains the same, $\lambda$ increases by 1.

The measure $2k - \lambda$ decreases in each step.

$\Rightarrow$ Height of the search tree $\leq 2k \leq 2^2k = 4k$ important cuts of size at most $k$. 
Important cuts

(2) Search tree algorithm for enumerating all these cuts:

An (arbitrary) edge \(uv\) leaving \(X = R_{\text{max}}\) is either in the cut or not.

Branch 1: If \(uv \in S\), then \(S \setminus uv\) is an important \((X, Y)\)-cut of size at most \(k - 1\) in \(G \setminus uv\).

Branch 2: If \(uv \notin S\), then \(S\) is an important \((X \cup v, Y)\)-cut of size at most \(k\) in \(G\).
Important cuts

(2) Search tree algorithm for enumerating all these cuts:
An (arbitrary) edge $uv$ leaving $X = R_{\text{max}}$ is either in the cut or not.

Branch 1: If $uv \in S$, then $S \setminus uv$ is an important $(X, Y)$-cut of size at most $k - 1$ in $G \setminus uv$.

$\Rightarrow$ $k$ decreases by one, $\lambda$ decreases by at most $1$.

Branch 2: If $uv \notin S$, then $S$ is an important $(X \cup v, Y)$-cut of size at most $k$ in $G$.

$\Rightarrow$ $k$ remains the same, $\lambda$ increases by $1$. 
Important cuts

(2) Search tree algorithm for enumerating all these cuts:
An (arbitrary) edge $uv$ leaving $X = R_{\text{max}}$ is either in the cut or not.

Branch 1: If $uv \in S$, then $S \setminus uv$ is an important $(X, Y)$-cut of size at most $k - 1$ in $G \setminus uv$.

$\Rightarrow k$ decreases by one, $\lambda$ decreases by at most 1.

Branch 2: If $uv \notin S$, then $S$ is an important $(X \cup v, Y)$-cut of size at most $k$ in $G$.

$\Rightarrow k$ remains the same, $\lambda$ increases by 1.

The measure $2k - \lambda$ decreases in each step.

$\Rightarrow$ Height of the search tree $\leq 2k$

$\Rightarrow \leq 2^{2k} = 4^k$ important cuts of size at most $k$. 
There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$ and they can be enumerated in time $O(4^k \cdot k \cdot (|V(G)| + |E(G)|))$.

Algorithm for enumerating important cuts:

1. Handle trivial cases ($k = 0$, $\lambda = 0$, $k < \lambda$)
2. Find $R_{\text{max}}$.
3. Choose an edge $uv$ of $\delta(R_{\text{max}})$.
   - Recurse on $(G - uv, R_{\text{max}}, Y, k - 1)$.
   - Recurse on $(G, R_{\text{max}} \cup v, Y, k)$.
4. Check if the returned cuts are important and throw away those that are not.
Important cuts

**Theorem**

There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

**Example:** The bound $4^k$ is essentially tight.
Important cuts

**Theorem**

There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

**Example:** The bound $4^k$ is essentially tight.

Any subtree with $k$ leaves gives an important $(X, Y)$-cut of size $k$. 
Important cuts

**Theorem**

There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

**Example:** The bound $4^k$ is essentially tight.

Any subtree with $k$ leaves gives an important $(X, Y)$-cut of size $k$. 
Important cuts

**Theorem**

There are at most $4^k$ important $(X, Y)$-cuts of size at most $k$.

**Example:** The bound $4^k$ is essentially tight.

Any subtree with $k$ leaves gives an important $(X, Y)$-cut of size $k$. The number of subtrees with $k$ leaves is the Catalan number

$$C_{k-1} = \frac{1}{k} \binom{2k - 2}{k - 1} \geq 4^k / \text{poly}(k).$$
**Multiway Cut**

**Definition:** A multiway cut of a set of terminals $T$ is a set $S$ of edges such that each component of $G \setminus S$ contains at most one vertex of $T$.

**Multiway Cut**

**Input:** Graph $G$, set $T$ of vertices, integer $k$

**Find:** A multiway cut $S$ of at most $k$ edges.

Polynomial for $|T| = 2$, but NP-hard for any fixed $|T| \geq 3$.

$\Rightarrow$ Cannot be FPT parameterized by $|T|$ assuming P $\neq$ NP.
**Multiway Cut**

**Definition:** A multiway cut of a set of terminals $T$ is a set $S$ of edges such that each component of $G \setminus S$ contains at most one vertex of $T$.

**Multiway Cut**

**Input:** Graph $G$, set $T$ of vertices, integer $k$

**Find:** A multiway cut $S$ of at most $k$ edges.

Trivial to solve in polynomial time for fixed $k$ (in time $n^{O(k)}$).

**Theorem**

Multiway Cut can be solved in time $4^k \cdot k^3 \cdot (|V(G)| + |E(G)|)$. 
Intuition: Consider a $t \in T$. A subset of the solution $S$ is a $(t, T \setminus t)$-cut.
Multiway Cut

**Intuition:** Consider a \( t \in T \). A subset of the solution \( S \) is a \((t, T \setminus t)\)-cut.

There are many such cuts.
**Multiway Cut**

**Intuition:** Consider a $t \in T$. A subset of the solution $S$ is a $(t, T \setminus t)$-cut.

There are many such cuts.
Multiway Cut

Intuition: Consider a $t \in T$. A subset of the solution $S$ is a $(t, T \setminus t)$-cut.

There are many such cuts.

But a cut farther from $t$ and closer to $T \setminus t$ seems to be more useful.
**Multiway Cut and important cuts**

**Pushing Lemma**

Let \( t \in T \). The *Multiway Cut* problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

**Proof:** Let \( R \) be the vertices reachable from \( t \) in \( G \setminus S \) for a solution \( S \). If \( \delta(R) \) is not important, then there is an important cut \( \delta(R') \) with \( R \subset R' \) and \(|\delta(R')| \leq |\delta(R)|\). Replace \( S \) with \( S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S| \)

\( S' \) is a multiway cut: (1) There is no \( t \)-\( u \) path in \( G \setminus S' \) and (2) a \( u \)-\( v \) path in \( G \setminus S' \) implies a \( t \)-\( u \) path, a contradiction.
Multiway Cut and important cuts

Pushing Lemma
Let \( t \in T \). The Multiway Cut problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

Proof: Let \( R \) be the vertices reachable from \( t \) in \( G \setminus S \) for a solution \( S \).
Multiway Cut and important cuts

Pushing Lemma

Let \( t \in T \). The Multiway Cut problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

Proof: Let \( R \) be the vertices reachable from \( t \) in \( G \setminus S \) for a solution \( S \).

\( \delta(R) \) is not important, then there is an important cut \( \delta(R') \) with \( R \subset R' \) and \( |\delta(R')| \leq |\delta(R)| \). Replace \( S \) with \( S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S| \)
Multiway Cut and important cuts

Pushing Lemma

Let \( t \in T \). The Multiway Cut problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

Proof: Let \( R \) be the vertices reachable from \( t \) in \( G \setminus S \) for a solution \( S \).

\[ \delta(R) \] is not important, then there is an important cut \( \delta(R') \) with \( R \subset R' \) and \(|\delta(R')| \leq |\delta(R)|\). Replace \( S \) with \( S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S| \)

\( S' \) is a multiway cut: (1) There is no \( t-u \) path in \( G \setminus S' \) and (2) a \( u-v \) path in \( G \setminus S' \) implies a \( t-u \) path, a contradiction.
Multiway Cut and important cuts

Pushing Lemma

Let \( t \in T \). The Multiway Cut problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

Proof: Let \( R \) be the vertices reachable from \( t \) in \( G \setminus S \) for a solution \( S \).

\( \delta(R) \) is not important, then there is an important cut \( \delta(R') \) with \( R \subset R' \) and \( |\delta(R')| \leq |\delta(R)| \). Replace \( S \) with \( S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S| \)

\( S' \) is a multiway cut: (1) There is no \( t-u \) path in \( G \setminus S' \) and (2) a \( u-v \) path in \( G \setminus S' \) implies a \( t-u \) path, a contradiction.
Algorithm for **Multiway Cut**

1. If every vertex of $T$ is in a different component, then we are done.
2. Let $t \in T$ be a vertex that is not separated from every $T \setminus t$.
3. Enumerate every important $(t, T \setminus t)$ cut of size at most $k$ and branch on choosing one such cut $S$.
4. Set $G := G \setminus S$ and $k := k - |S|$.
5. Go to step 1.

We branch into at most $4^k$ directions at most $k$ times: $4^{k^2} \cdot n^{O(1)}$ running time.

**Next:** Better analysis gives $4^k$ bound on the size of the search tree.
A refined bound

We have seen: at most $4^k$ important cut of size at most $k$.

Better bound:

**Lemma**

If $S$ is the set of all important $(X, Y)$-cuts, then $\sum_{S \in S} 4^{-|S|} \leq 1$ holds.
A refined bound

We have seen: at most $4^k$ important cut of size at most $k$.

Better bound:

**Lemma**

If $S$ is the set of all important $(X, Y)$-cuts, then $\sum_{S \in S} 4^{-|S|} \leq 1$ holds.

Better algorithm:

**Lemma**

We can enumerate the set $S_k$ of every important $(X, Y)$-cut of size at most $k$ in time $O(|S_k| \cdot k^2 \cdot (|V(G)| + |E(G)|))$. 
Refined analysis for **Multiway Cut**

**Lemma**

If $S$ is the set of all important $(X, Y)$-cuts, then $\sum_{S \in S} 4^{-|S|} \leq 1$ holds.

**Lemma**

The search tree for the **Multiway Cut** algorithm has $4^k$ leaves.

**Proof:** Let $L_k$ be the maximum number of leaves with parameter $k$. We prove $L_k \leq 4^k$ by induction. After enumerating the set $S_k$ of important cuts of size $\leq k$, we branch into $|S_k|$ directions.

$$\sum_{S \in S_k} 4^{k - |S|} = 4^k \cdot \sum_{S \in S_k} 4^{-|S|} \leq 4^k$$
Algorithm for Multiway Cut

Theorem
Multiway Cut can be solved in time $O(4^k \cdot k^3 \cdot (|V(G)| + |E(G)|))$.

1. If every vertex of $T$ is in a different component, then we are done.
2. Let $t \in T$ be a vertex that is not separated from every $T \setminus t$.
3. Enumerate every important $(t, T \setminus t)$ cut of size at most $k$ and branch on choosing one such cut $S$.
4. Set $G := G \setminus S$ and $k := k - |S|$.
5. Go to step 1.
**Multicut**

**Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i$-$t_i$ path for any $i$.

---

**Theorem**

**Multicut** can be solved in time $f(k, \ell) \cdot n^{O(1)}$ (FPT parameterized by combined parameters $k$ and $\ell$).
**Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i-t_i$ path for any $i$.

---

**Theorem**

**Multicut** can be solved in time $f(k, \ell) \cdot n^{O(1)}$ (FPT parameterized by combined parameters $k$ and $\ell$).

**Proof:** The solution partitions $\{s_1, t_1, \ldots, s_\ell, t_\ell\}$ into components. Guess this partition, contract the vertices in a class, and solve **Multiway Cut**.

---

**Theorem**

**Multicut** is FPT parameterized by the size $k$ of the solution.
Important cuts

Definition

A minimal \((X, Y)\)-cut \(\delta(R)\) is **important** if there is no \((X, Y)\)-cut \(\delta(R')\) with \(R \subset R'\) and \(|\delta(R')| \leq |\delta(R)|\).
Directed graphs

**Definition:** $\vec{\delta}(R)$ is the set of edges leaving $R$.

**Observation:** Every inclusionwise-minimal directed $(X, Y)$-cut $S$ can be expressed as $S = \vec{\delta}(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.

**Definition:** A minimal $(X, Y)$-cut $\vec{\delta}(R)$ is **important** if there is no $(X, Y)$-cut $\vec{\delta}(R')$ with $R \subset R'$ and $|\vec{\delta}(R')| \leq |\vec{\delta}(R)|$. 
Directed graphs

**Definition:** $\vec{\delta}(R)$ is the set of edges leaving $R$.

**Observation:** Every inclusionwise-minimal directed $(X, Y)$-cut $S$ can be expressed as $S = \vec{\delta}(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.

**Definition:** A minimal $(X, Y)$-cut $\vec{\delta}(R)$ is **important** if there is no $(X, Y)$-cut $\vec{\delta}(R')$ with $R \subset R'$ and $|\vec{\delta}(R')| \leq |\vec{\delta}(R)|$.
Directed graphs

**Definition:** $\vec{\delta}(R)$ is the set of edges leaving $R$.

**Observation:** Every inclusionwise-minimal directed $(X, Y)$-cut $S$ can be expressed as $S = \vec{\delta}(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.

**Definition:** A minimal $(X, Y)$-cut $\vec{\delta}(R)$ is **important** if there is no $(X, Y)$-cut $\vec{\delta}(R')$ with $R \subset R'$ and $|\vec{\delta}(R')| \leq |\vec{\delta}(R)|$.

The proof for the undirected case goes through for the directed case:

**Theorem**

There are at most $4^k$ important directed $(X, Y)$-cuts of size at most $k$. 
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)

Let \( t \in T \). The Multiway Cut problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

Directed counterexample:

![Diagram](image)

Unique solution with \( k = 1 \) edges, but it is not an important cut (boundary of \( \{s, a\} \), but the boundary of \( \{s, a, b\} \) has same size).
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)
Let $t \in T$. The Multiway Cut problem has a solution $S$ that contains an important $(t, T \setminus t)$-cut.

Directed counterexample:

Unique solution with $k = 1$ edges, but it is not an important cut (boundary of $\{s, a\}$, but the boundary of $\{s, a, b\}$ has same size).
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)
Let $t \in T$. The Multiway Cut problem has a solution $S$ that contains an important $(t, T \setminus t)$-cut.

Directed counterexample:

Unique solution with $k = 1$ edges, but it is not an important cut (boundary of $\{s, a\}$, but the boundary of $\{s, a, b\}$ has same size).
The undirected approach does not work: the pushing lemma is not true.

**Pushing Lemma (for undirected graphs)**

Let $t \in T$. The **Multiway Cut** problem has a solution $S$ that contains an important $(t, T \setminus t)$-cut.

**Problem in the undirected proof:**

Replacing $R$ by $R'$ cannot create a $t \rightarrow u$ path, but can create a $u \rightarrow t$ path.
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

**Pushing Lemma (for undirected graphs)**

Let \( t \in T \). The **Multiway Cut** problem has a solution \( S \) that contains an important \((t, T \setminus t)\)-cut.

Using additional techniques, one can show:

**Theorem**

**Directed Multiway Cut** is FPT parameterized by the size \( k \) of the solution.
Directed Multicut

**Directed Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i \rightarrow t_i$ path for any $i$.

**Theorem**

**Directed Multicut** with $\ell = 4$ is W[1]-hard parameterized by $k$. 
**Directed Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i \rightarrow t_i$ path for any $i$.

**Theorem**

**Directed Multicut** with $\ell = 4$ is W[1]-hard parameterized by $k$.

But the case $\ell = 2$ can be reduced to **Directed Multiway Cut**:

![Diagram](https://via.placeholder.com/150)
Directed Multicut

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i \rightarrow t_i$ path for any $i$.

**Theorem**

Direct Multicut with $\ell = 4$ is W[1]-hard parameterized by $k$.

But the case $\ell = 2$ can be reduced to Directed Multiway Cut:
**Directed Multicut**

**Directed Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i \rightarrow t_i$ path for any $i$.

---

**Theorem**

**Directed Multicut** with $\ell = 4$ is W[1]-hard parameterized by $k$.

But the case $\ell = 2$ can be reduced to **Directed Multiway Cut**:

![Diagram of Directed Multicut and Multiway Cut](image-url)
**Directed Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of edges such that $G \setminus S$ has no $s_i \rightarrow t_i$ path for any $i$.

---

**Theorem**

**Directed Multicut** with $\ell = 4$ is W[1]-hard parameterized by $k$.

**Corollary**

**Directed Multicut** with $\ell = 2$ is FPT parameterized by the size $k$ of the solution.

---

**Open:** Is **Directed Multicut** with $\ell = 3$ FPT?
**Skew Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of $k$ directed edges such that $G \setminus S$ contains no $s_i \to t_j$ path for any $i \geq j$. 

![Diagram of Skew Multicut](image-url)
**Skew Multicut**

**Skew Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of $k$ directed edges such that $G \setminus S$ contains no $s_i \to t_j$ path for any $i \geq j$.

---

**Pushing Lemma**

**Skew Multicut** problem has a solution $S$ that contains an important $(s_\ell, \{t_1, \ldots, t_\ell\})$-cut.
**Skew Multicut**

**Input:** Graph $G$, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer $k$

**Find:** A set $S$ of $k$ directed edges such that $G \setminus S$ contains no $s_i \rightarrow t_j$ path for any $i \geq j$.

**Theorem**

**Skew Multicut** can be solved in time $4^k \cdot n^{O(1)}$. 
Directed Feedback Vertex Set

**Directed Feedback Vertex/Edge Set**

**Input:** Directed graph $G$, integer $k$

**Find:** A set $S$ of $k$ vertices/edges such that $G \setminus S$ is acyclic.

**Note:** Edge and vertex versions are equivalent, we will consider the edge version here.

**Note:** It is not a generalization of *(Undirected)* Feedback Vertex Set!

**Theorem**

Directed Feedback Edge Set is FPT parameterized by the size $k$ of the solution.

Solution uses the technique of iterative compression.
The compression problem

**Directed Feedback Edge Set Compression**

**Input:** Directed graph $G$, integer $k$,
   a set $W$ of $k + 1$ edges such that $G \setminus W$ is acyclic

**Find:** A set $S$ of $k$ edges such that $G \setminus S$ is acyclic.

Easier than the original problem, as the extra input $W$ gives us useful structural
information about $G$.

**Lemma**

The compression problem is FPT parameterized by $k$. 
The compression problem

**Directed Feedback Edge Set Compression**

**Input:** Directed graph $G$, integer $k$,

a set $W$ of $k + 1$ vertices such that $G \setminus W$ is acyclic

**Find:** A set $S$ of $k$ edges such that $G \setminus S$ is acyclic.

Easier than the original problem, as the extra input $W$ gives us useful structural information about $G$.

**Lemma**

The compression problem is FPT parameterized by $k$.

A useful trick for edge deletion problems: we define the compression problem in a way that a solution of $k + 1$ vertices are given and we have to find a solution of $k$ edges.
The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each $w_i$ into an edge $t_is_i$.

By guessing the order of $\{w_1, \ldots, w_{k+1}\}$ in the acyclic ordering of $G \setminus S$, we can assume that $w_1 < w_2 < \cdots < w_{k+1}$ in $G \setminus S \ [(k+1)! \text{ possibilities}].$
The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each $w_i$ into an edge $t_is_i$.

Claim:

$G \setminus S$ is acyclic and has an ordering with $w_1 < w_2 < \cdots < w_{k+1}$

$\Downarrow$

$S$ covers every $s_i \rightarrow t_j$ path for every $i \geq j$

$\Downarrow$

$G \setminus S$ is acyclic
The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each $w_i$ into an edge $t_i s_i$.

Claim:

$G \setminus S$ is acyclic and has an ordering with $w_1 < w_2 < \cdots < w_{k+1}$

$\Downarrow$

$S$ covers every $s_i \rightarrow t_j$ path for every $i \geq j$

$\Downarrow$

$G \setminus S$ is acyclic
The compression problem

Proof: Let \( W = \{w_1, \ldots, w_{k+1}\} \)
Let us split each \( w_i \) into an edge \( t_is_i \).

Claim:

\( G \setminus S \) is acyclic and has an ordering with \( w_1 < w_2 < \cdots < w_{k+1} \)
\( \Downarrow \)
\( S \) covers every \( s_i \rightarrow t_j \) path for every \( i \geq j \)
\( \Downarrow \)
\( G \setminus S \) is acyclic
Iterative compression

We have given a $f(k)n^{O(1)}$ algorithm for the following problem:

**Directed Feedback Edge Set Compression**

**Input:** Directed graph $G$, integer $k$, a set $W$ of $k + 1$ vertices such that $G \setminus W$ is acyclic

**Find:** A set $S$ of $k$ edges such that $G \setminus S$ is acyclic.

Nice, but how do we get a solution $W$ of size $k + 1$?
Iterative compression

We have given a $f(k)n^{O(1)}$ algorithm for the following problem:

**Directed Feedback Edge Set Compression**

**Input:** Directed graph $G$, integer $k$,
- a set $W$ of $k + 1$ vertices such that $G \setminus W$ is acyclic

**Find:** A set $S$ of $k$ edges such that $G \setminus S$ is acyclic.

Nice, but how do we get a solution $W$ of size $k + 1$?

**We get it for free!**

Powerful technique: *iterative compression.*
Iterative compression

Let $v_1, \ldots, v_n$ be the vertices of $G$ and let $G_i$ be the subgraph induced by $\{v_1, \ldots, v_i\}$. For every $i = 1, \ldots, n$, we find a set $S_i$ of at most $k$ edges such that $G_i \setminus S_i$ is acyclic.
Iterative compression

Let \( v_1, \ldots, v_n \) be the vertices of \( G \) and let \( G_i \) be the subgraph induced by \{\( v_1, \ldots, v_i \}\). For every \( i = 1, \ldots, n \), we find a set \( S_i \) of at most \( k \) edges such that \( G_i \setminus S_i \) is acyclic.

- For \( i = 1 \), we have the trivial solution \( S_i = \emptyset \).
Iterative compression

Let $v_1, \ldots, v_n$ be the vertices of $G$ and let $G_i$ be the subgraph induced by $\{v_1, \ldots, v_i\}$. For every $i = 1, \ldots, n$, we find a set $S_i$ of at most $k$ edges such that $G_i \setminus S_i$ is acyclic.

- For $i = 1$, we have the trivial solution $S_i = \emptyset$.
- Suppose we have a solution $S_i$ for $G_i$. Let $W_i$ contain the head of each edge in $S_i$. Then $W_i \cup \{v_{i+1}\}$ is a set of at most $k + 1$ vertices whose removal makes $G_{i+1}$ acyclic.
Iterative compression

Let $v_1, \ldots, v_n$ be the vertices of $G$ and let $G_i$ be the subgraph induced by $\{v_1, \ldots, v_i\}$. For every $i = 1, \ldots, n$, we find a set $S_i$ of at most $k$ edges such that $G_i \setminus S_i$ is acyclic.

- For $i = 1$, we have the trivial solution $S_i = \emptyset$.
- Suppose we have a solution $S_i$ for $G_i$. Let $W_i$ contain the head of each edge in $S_i$. Then $W_i \cup \{v_{i+1}\}$ is a set of at most $k + 1$ vertices whose removal makes $G_{i+1}$ acyclic.
- Use the compression algorithm for $G_{i+1}$ with the set $W_i \cup \{v_{i+1}\}$.
  - If there is no solution of size $k$ for $G_{i+1}$, then we can stop.
  - Otherwise the compression algorithm gives a solution $S_{i+1}$ of size $k$ for $G_{i+1}$.

Running time: We call the compression algorithm $n$ times, everything else is polynomial.

Theorem

Directed Feedback Edge Set is FPT parameterized by the size $k$ of the solution.
Iterative compression

Let $v_1, \ldots, v_n$ be the vertices of $G$ and let $G_i$ be the subgraph induced by $\{v_1, \ldots, v_i\}$. For every $i = 1, \ldots, n$, we find a set $S_i$ of at most $k$ edges such that $G_i \setminus S_i$ is acyclic.

- For $i = 1$, we have the trivial solution $S_i = \emptyset$.
- Suppose we have a solution $S_i$ for $G_i$. Let $W_i$ contain the head of each edge in $S_i$. Then $W_i \cup \{v_{i+1}\}$ is a set of at most $k + 1$ vertices whose removal makes $G_{i+1}$ acyclic.
- Use the compression algorithm for $G_{i+1}$ with the set $W_i \cup \{v_{i+1}\}$.
  - If there is no solution of size $k$ for $G_{i+1}$, then we can stop.
  - Otherwise the compression algorithm gives a solution $S_{i+1}$ of size $k$ for $G_{i+1}$.

Running time: We call the compression algorithm $n$ times, everything else is polynomial.

Theorem

**Directed Feedback Edge Set** is FPT parameterized by the size $k$ of the solution.
Summary

- Definition of important cuts.
- Simple but essentially tight combinatorial bound on the number of important cuts.
- Pushing argument: we can assume that the solution contains an important cut. Solves Multiway Cut, Skew Multicut.
- Iterative compression reduces Directed Feedback Edge Set to Skew Multicut.