Important cuts

Dániel Marx

Lecture #9
July 3, 2020

Overview

Main message

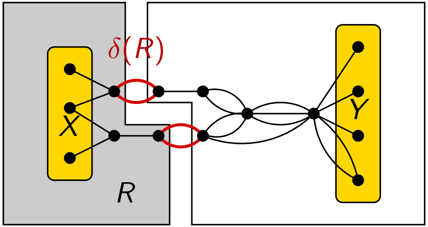
Small cuts in graphs have interesting extremal properties that can be exploited in combinatorial and algorithmic results.

- 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

Minimum 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 $R \cap Y = \emptyset$.



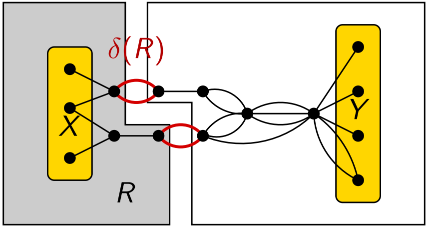
Minimum cuts

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

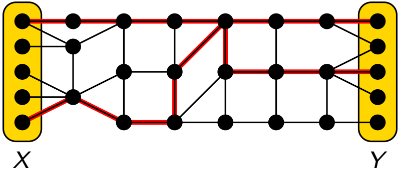
An (X, Y)-cut of size at most k (if exists) can be found in time $O(k \cdot (|V(G)| + |E(G)|))$.

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

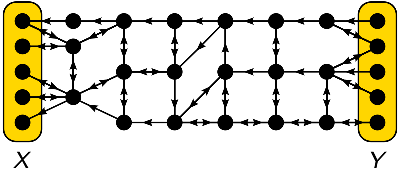
Residual graph:

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

original graph



residual graph



Finding minimum cuts

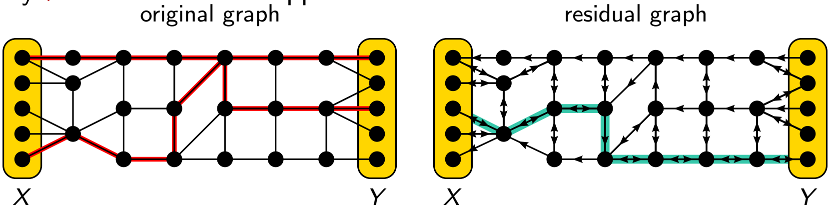
Theorem

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

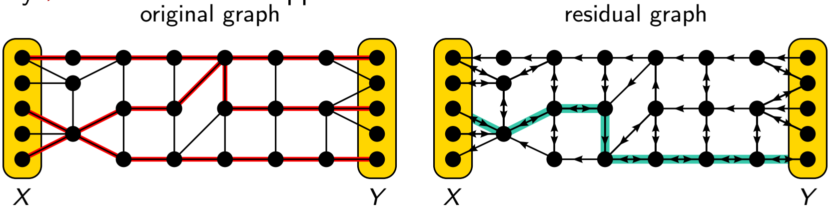
Theorem

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.



Finding minimum cuts

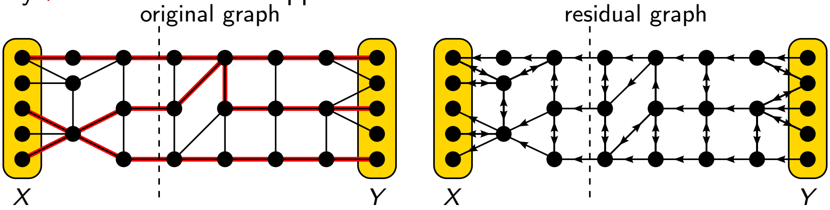
Theorem

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 δ is **submodular:** for arbitrary sets A, B,

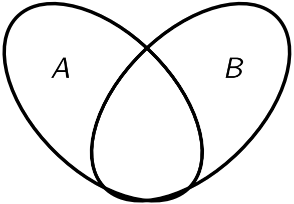
$$|\delta(A)| \quad + \quad |\delta(B)| \quad \geq \quad |\delta(A \cap B)| \quad + \quad |\delta(A \cup B)|$$

Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

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

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

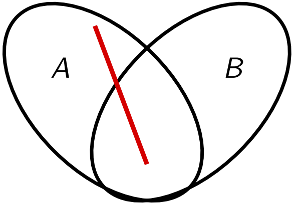


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underset{0}{|\delta(A)|} + \underset{1}{|\delta(B)|} \geq \underset{1}{|\delta(A \cap B)|} + \underset{0}{|\delta(A \cup B)|}$$

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

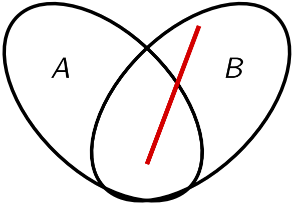


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

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

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

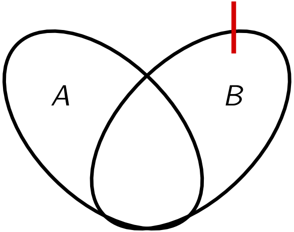


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

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

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

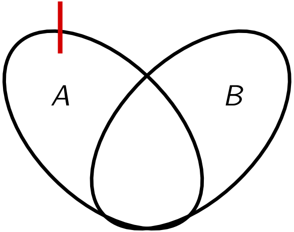


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

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

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

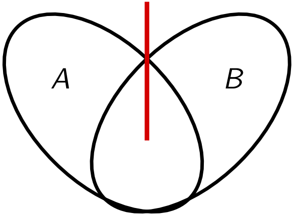


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

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

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

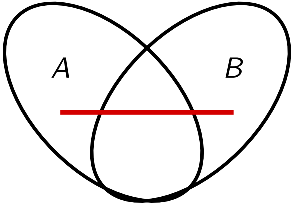


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

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

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



Submodularity

Lemma

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

Submodularity

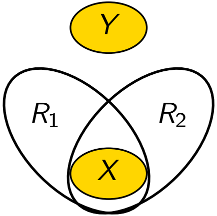
Lemma

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

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 λ.

$$\underset{\lambda}{|\delta(R_1)|} + \underset{\lambda}{|\delta(R_2)|} \geq \underset{\geq \lambda}{|\delta(R_1 \cap R_2)|} + |\delta(R_1 \cup R_2)|$$

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



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

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

Lemma

Given a graph G and sets $X, Y \subseteq V(G)$, the sets $R_{\min}$ and $R_{\max}$ 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, $X = R_{\max}$. Similar for $R_{\min}$.

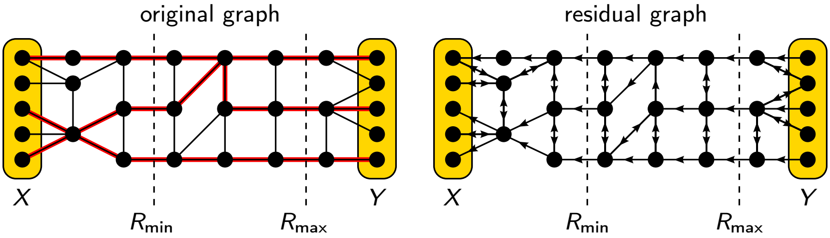
But we can do better!

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

Lemma

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

Proof: Look at the residual graph.



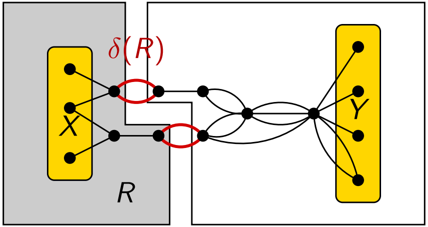
$R_{\min}$: vertices reachable from X.
$R_{\max}$: vertices from which Y is **not** reachable.

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 $R \cap Y = \emptyset$.

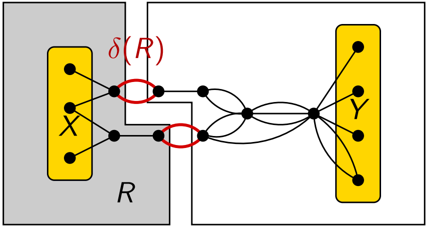


Important cuts

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.

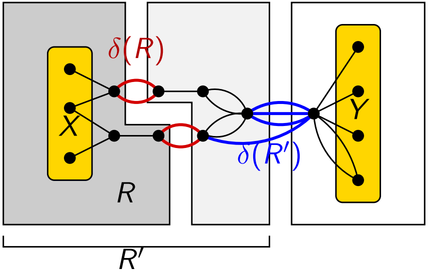


Important cuts

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.

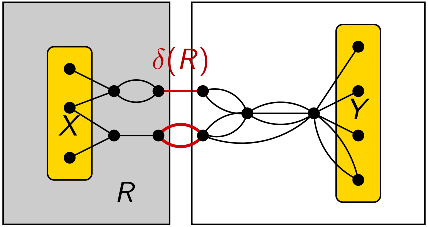


Important cuts

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.

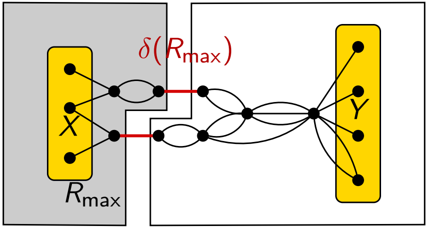


Important cuts

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.

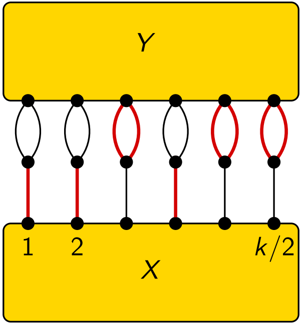


Observation: There is a unique important (X, Y)-cut of minimum size: $\delta(R_{\max})$.

Important cuts

The number of important cuts can be exponentially large.

Example:

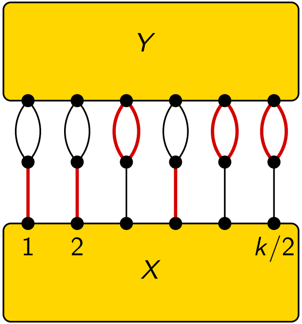


This graph has $2^{k/2}$ important (X, Y)-cuts of size at most k.

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

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

Important cuts

Theorem

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

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

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

By the submodularity of δ:

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

$$\Downarrow$$

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

$$\Downarrow$$

If $R \neq R_{\max} \cup R$, 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 λ be the minimum (X, Y)-cut size and let $\delta(R_{\max})$ be the unique important cut of size λ such that $R_{\max}$ is maximal.

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

By the submodularity of δ:

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

$$\Downarrow$$

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

$$\Downarrow$$

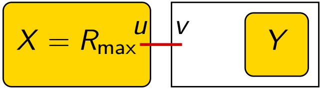
If $R \neq R_{\max} \cup R$, then $\delta(R)$ is not important.

Thus the important (X, Y)- and $(R_{\max}, Y)$-cuts are the same.
$\Rightarrow$ We can assume $X = R_{\max}$.

Important cuts

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

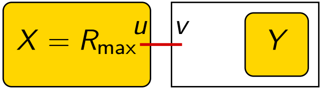
An (arbitrary) edge uv leaving $X = R_{\max}$ is either in the cut or not.



Important cuts

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

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



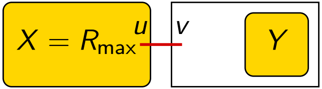
Branch 1: If $uv \in S$, then $S \setminus uv$ is an important (X, Y)-cut of size at most $k - 1$ in $G \setminus uv$.

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 $X = R_{\max}$ 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, λ decreases by at most 1.

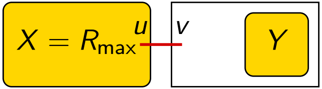
Branch 2: If $uv \notin S$, then S is an important $(X \cup v, Y)$-cut of size at most k in G.

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

Important cuts

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

An (arbitrary) edge uv leaving $X = R_{\max}$ 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, λ 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, λ 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.

Important cuts — some details

We are using the following two statements:

Branch 1: If $uv \in S$, then

S is an important (X, Y)-cut in G → $S \setminus uv$ is an important (X, Y)-cut in $G \setminus uv$

Branch 2: If S is an $(X \cup v, Y)$-cut, then

S is an important (X, Y)-cut in G → S is an important $(X \cup v, Y)$-cut in G

Important cuts — some details

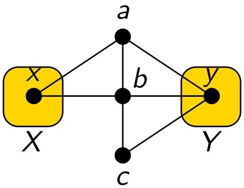
We are using the following two statements:

Branch 1: If $uv \in S$, then

S is an important (X, Y)-cut in G $\Rightarrow$ $\not\Leftarrow$ $S \setminus uv$ is an important (X, Y)-cut in $G \setminus uv$

Converse is not true:

Set $\{ab, ay\}$ is important (X, Y)-cut in $G \setminus xb$, but $\{xb, ab, ay\}$ is not an important (X, Y)-cut in G.



Branch 2: If S is an $(X \cup v, Y)$-cut, then

S is an important (X, Y)-cut in G $\Leftrightarrow$ S is an important $(X \cup v, Y)$-cut in G

Converse is true!

Important cuts — some details

We are using the following two statements:

Branch 1: If $uv \in S$, then

S is an important (X, Y)-cut in G

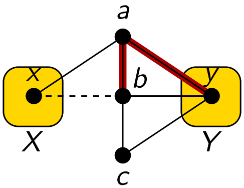
$\Rightarrow$

$\not\Leftarrow$

$S \setminus uv$ is an important (X, Y)-cut in $G \setminus uv$

Converse is not true:

Set $\{ab, ay\}$ is important (X, Y)-cut in $G \setminus xb$, but $\{xb, ab, ay\}$ is not an important (X, Y)-cut in G.



Branch 2: If S is an $(X \cup v, Y)$-cut, then

S is an important (X, Y)-cut in G

$\Leftrightarrow$

S is an important $(X \cup v, Y)$-cut in G

Converse is true!

Important cuts — algorithm

Theorem

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:

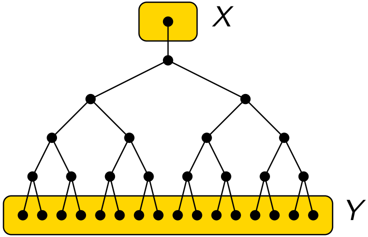
1. Handle trivial cases ($k = 0$, $\lambda = 0$, $k < \lambda$)
2. Find $R_{\max}$.
3. Choose an edge uv of $\delta(R_{\max})$.
    - Recurse on $(G - uv, R_{\max}, Y, k - 1)$.
    - Recurse on $(G, R_{\max} \cup v, Y, k)$.
4. Check if the returned cuts are important and throw away those that are not.

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.

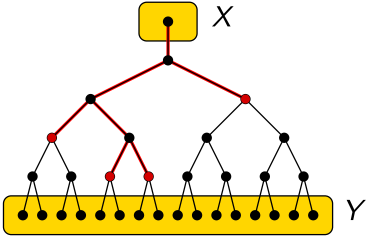


Important cuts

Theorem

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

Example: The bound 4^k is essentially tight.



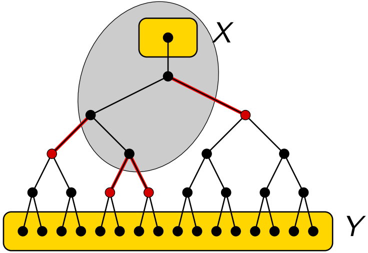
Any subtree with k leaves gives an important (X, Y)-cut of size k.

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.



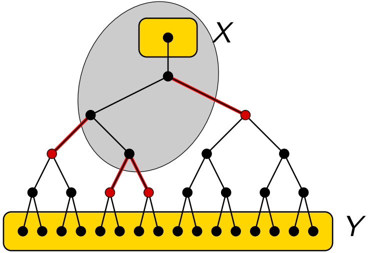
Any subtree with k leaves gives an important (X, Y)-cut of size k.

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

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

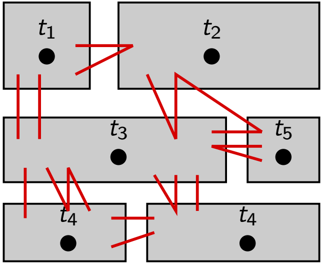
MULTIWAY CUT

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.

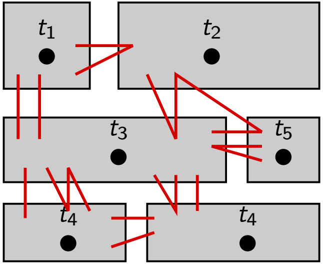


Polynomial for $|T| = 2$, but NP-hard for any fixed $|T| \geq 3$.

$\Rightarrow$ 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.

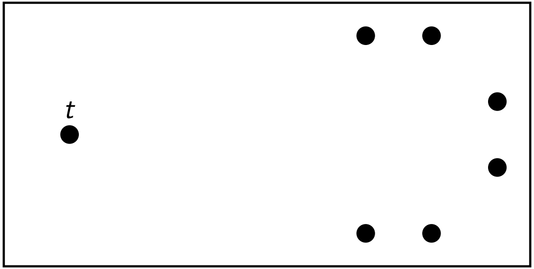
Trivial to solve in polynomial time for fixed k (in time $n^{O(k)}$).

Theorem

MULTIWAY CUT can be solved in time $4^k \cdot k^3 \cdot (|V(G)| + |E(G)|)$.

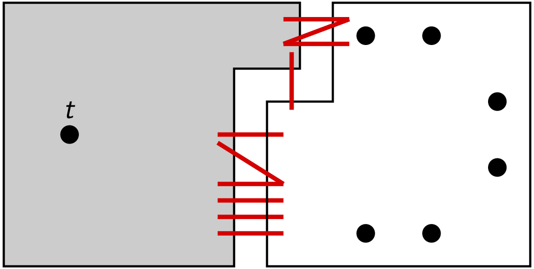
MULTIWAY CUT

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



MULTIWAY CUT

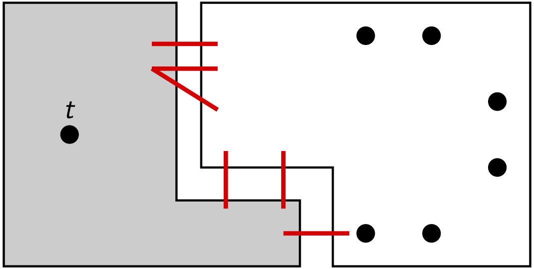
Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-cut.



There are many such cuts.

MULTIWAY CUT

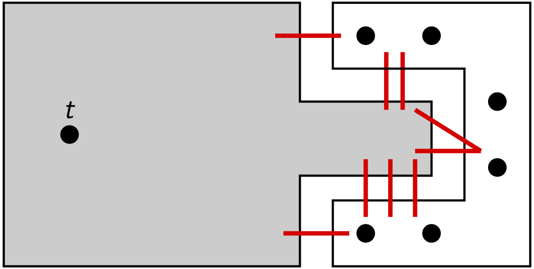
Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-cut.



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

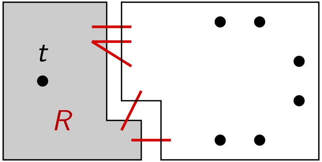
Let $t \in T$. The MULTIWAY CUT problem has a solution S that contains an important $(t, T \setminus t)$-cut.

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.

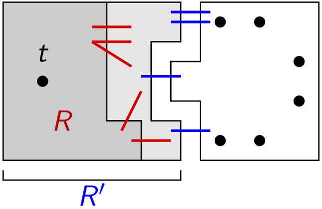


MULTIWAY CUT and important cuts

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.



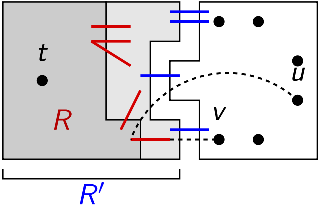
$\delta(R)$ is not important, then there is an important cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \leq |\delta(R)|$. Replace S with $S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S|$

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 $S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S|$

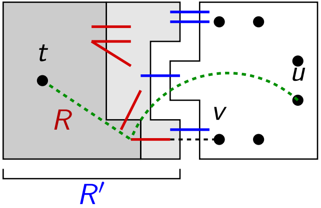
S' is a multiway cut: (1) There is no t-u path in $G \setminus S'$ and (2) a u-v path in $G \setminus S'$ implies a t-u path, a contradiction.

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

A refined bound

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

Better bound:

Lemma

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

Better algorithm:

Lemma

We can enumerate the set $\mathcal{S}_k$ of every important (X, Y)-cut of size at most k in time $O(|\mathcal{S}_k| \cdot k^2 \cdot (|V(G)| + |E(G)|))$.

Refined analysis for MULTIWAY CUT

Lemma

If $\mathcal{S}$ is the set of all important (X, Y)-cuts, then $\sum_{S\in\mathcal{S}} 4^{-|S|} \leq 1$ 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 $\mathcal{S}_k$ of important cuts of size $\leq k$, we branch into $|S_k|$ directions.

$$\sum_{S\in\mathcal{S}_k} 4^{k-|S|} = 4^k \cdot \sum_{S\in\mathcal{S}_k} 4^{-|S|} \leq 4^k$$

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 imporant $(t, T \setminus t)$ 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), $\dots$, (s_ℓ, t_ℓ), 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 ℓ).

MULTICUT

MULTICUT

Input: Graph G, pairs (s_1, t_1), ..., (s_ℓ, t_ℓ), 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 ℓ).

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.

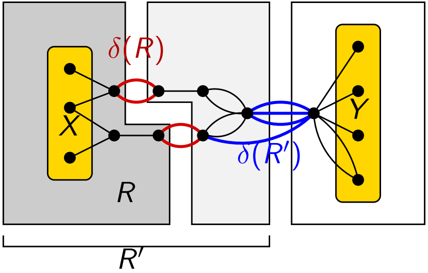
Theorem

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 $|\delta(R')| \leq |\delta(R)|$.



Simple combinatorial bound

Lemma:

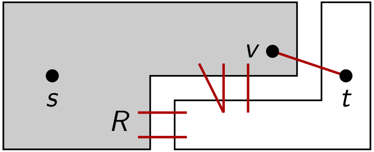
At most $k \cdot 4^k$ edges incident to t can be part of an inclusionwise minimal $s - t$ cut of size at most k.

Simple combinatorial bound

Lemma:

At most $k \cdot 4^k$ edges incident to t can be part of an inclusionwise minimal $s - t$ cut of size at most k.

Proof: We show that every such edge is contained in an important (s,t)-cut of size at most k.



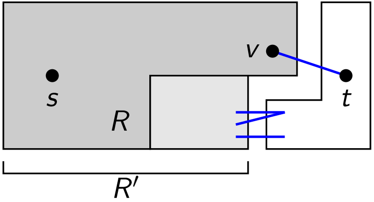
Suppose that $vt \in \delta(R)$ and $|\delta(R)| = k$.

Simple combinatorial bound

Lemma:

At most $k \cdot 4^k$ edges incident to t can be part of an inclusionwise minimal $s - t$ cut of size at most k.

Proof: We show that every such edge is contained in an important (s,t)-cut of size at most k.

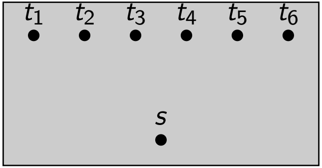


Suppose that $vt \in \delta(R)$ and $|\delta(R)| = k$.
There is an important (s,t)-cut $\delta(R')$ with $R \subseteq R'$ and $|\delta(R')| \leq k$.
Clearly, $vt \in \delta(R')$: $v \in R$, hence $v \in R'$.

Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

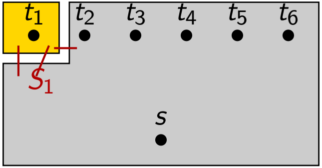
It is possible that n is "large" even if k is "small."



Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

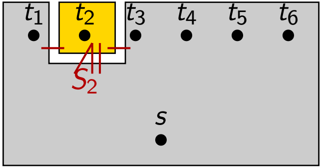
It is possible that n is "large" even if k is "small."



Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

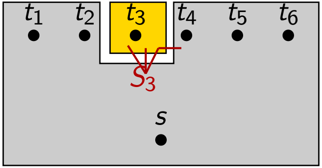
It is possible that n is "large" even if k is "small."



Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

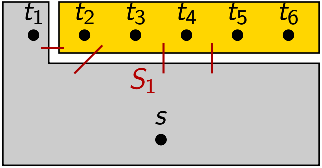
It is possible that n is "large" even if k is "small."



Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

It is possible that n is "large" even if k is "small."

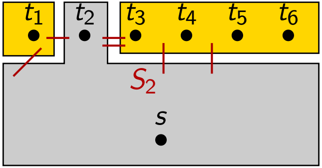


Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

It is possible that n is "large" even if k is "small."

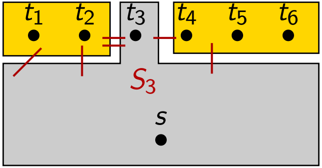


Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

It is possible that n is "large" even if k is "small."

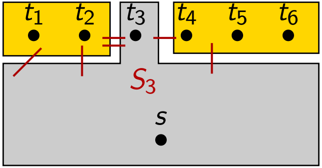


Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Anti isolation

Let $s, t_1, \ldots, t_n$ be vertices and $S_1, \ldots, S_n$ be sets of at most k edges such that S_i separates t_i from s, but S_i **does not** separate t_j from s for any $j \neq i$.

It is possible that n is "large" even if k is "small."

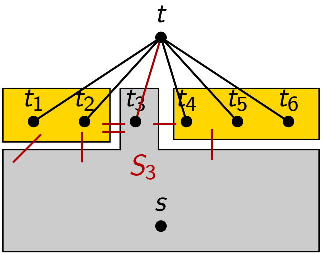


Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Lemma

If S_i separates t_j from s if and only $j \neq i$ and every S_i has size at most k, then $n \leq (k+1) \cdot 4^{k+1}$.

Anti isolation



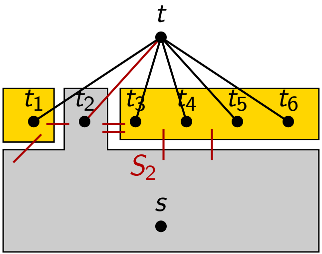
Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Lemma

If S_i separates t_j from s if and only $j \neq i$ and every S_i has size at most k, then $n \leq (k+1) \cdot 4^{k+1}$.

Proof: Add a new vertex t. Every edge tt_i is part of an (inclusionwise minimal) (s,t)-cut of size at most $k+1$. Use the previous lemma.

Anti isolation



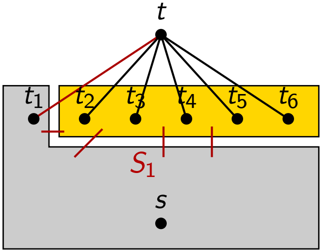
Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Lemma

If S_i separates t_j from s if and only $j \neq i$ and every S_i has size at most k, then $n \leq (k+1) \cdot 4^{k+1}$.

Proof: Add a new vertex t. Every edge tt_i is part of an (inclusionwise minimal) (s,t)-cut of size at most $k+1$. Use the previous lemma.

Anti isolation



Is the opposite possible, i.e., S_i separates every t_j **except** t_i?

Lemma

If S_i separates t_j from s if and only $j \neq i$ and every S_i has size at most k, then $n \leq (k+1) \cdot 4^{k+1}$.

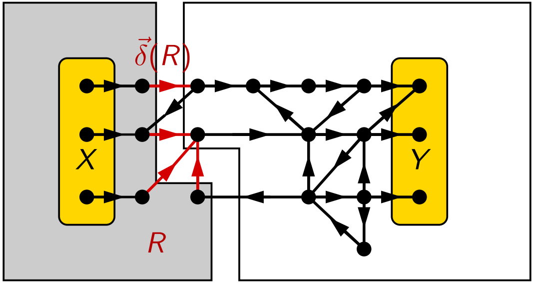
Proof: Add a new vertex t. Every edge tt_i is part of an (inclusionwise minimal) (s,t)-cut of size at most $k+1$. Use the previous lemma.

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 $|\vec{\delta}(R')| \leq |\vec{\delta}(R)|$.

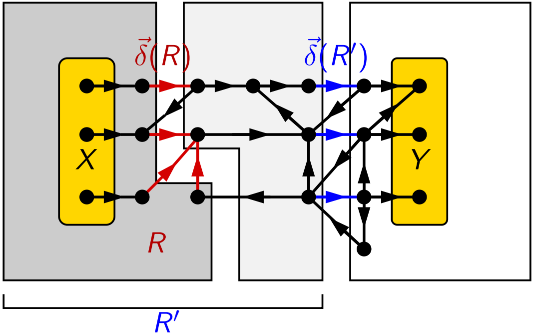


Directed graphs

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 $|\vec{\delta}(R')| \le |\vec{\delta}(R)|$.



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.

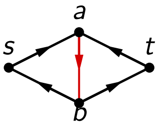
DIRECTED MULTIWAY CUT

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:



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).

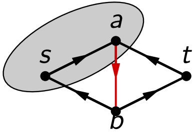
DIRECTED MULTIWAY CUT

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).

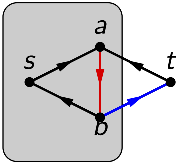
DIRECTED MULTIWAY CUT

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).

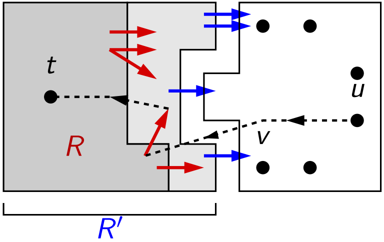
DIRECTED MULTIWAY CUT

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.

Problem in the undirected proof:



Replacing R by R' cannot create a $t \to u$ path, but can create a $u \to t$ 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 $s_i \to t_i$ path for any i.

Theorem

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

DIRECTED MULTICUT

DIRECTED MULTICUT

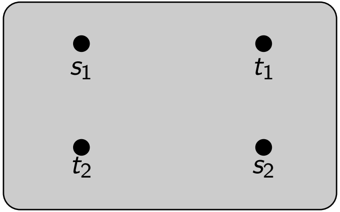
Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ 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:



DIRECTED MULTICUT

DIRECTED MULTICUT

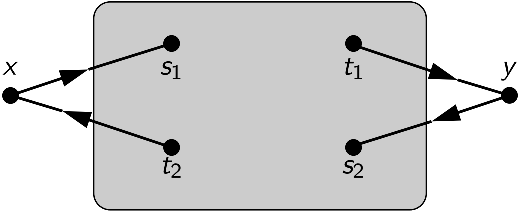
Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ 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:



DIRECTED MULTICUT

DIRECTED MULTICUT

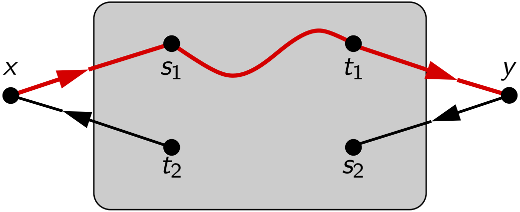
Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ 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:



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 $s_i \to t_i$ 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.

?

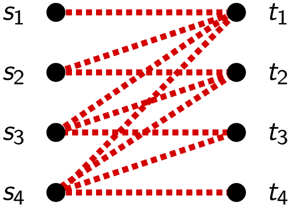
Open: Is DIRECTED MULTICUT with $\ell = 3$ FPT?

SKEW MULTICUT

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 $i \geq j$.

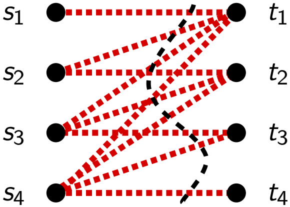


SKEW MULTICUT

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

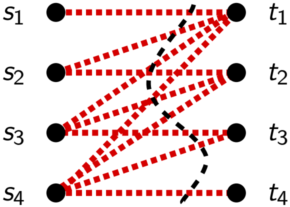
SKEW MULTCUT problem has a solution S that contains an important $(s_\ell, \{t_1, \ldots, t_\ell\})$-cut.

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 $i \geq j$.



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.

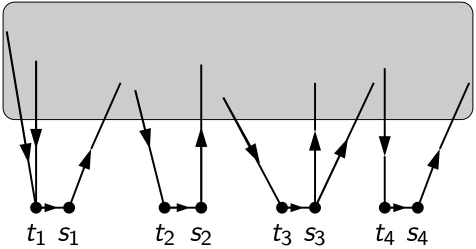
Lemma

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

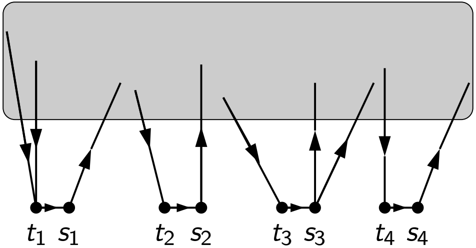
Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each w_i into an edge $\overrightarrow{t_i s_i}$.



- 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 $G \setminus S$ $[(k+1)!$ possibilities$]$.

The compression problem

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

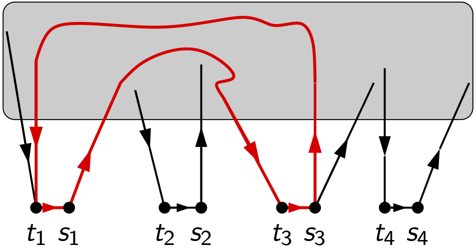


Claim:

$$G \setminus S \text{ is acyclic and has an ordering with } w_1 < w_2 < \cdots < w_{k+1}$$
$$\Downarrow$$
$$S \text{ covers every } s_i \to t_j \text{ path for every } i \geq j$$
$$\Downarrow$$
$$G \setminus S \text{ is acyclic}$$

The compression problem

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



Claim:

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

$$\Downarrow$$

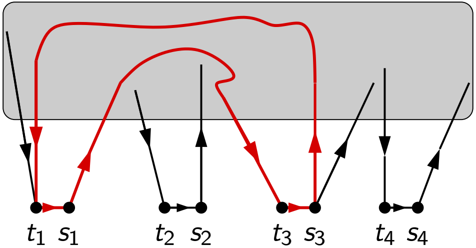
$$S \text{ covers every } s_i \to t_j \text{ path for every } i \geq j$$

$$\Downarrow$$

$$G \setminus S \text{ is acyclic}$$

The compression problem

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



Claim:

$$G \setminus S \text{ is acyclic and has an ordering with } w_1 < w_2 < \cdots < w_{k+1}$$
$$\Downarrow$$
$$S \text{ covers every } s_i \to t_j \text{ path for every } i \geq j$$
$$\Downarrow$$
$$G \setminus S \text{ is acyclic}$$

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, ..., 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}.

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.