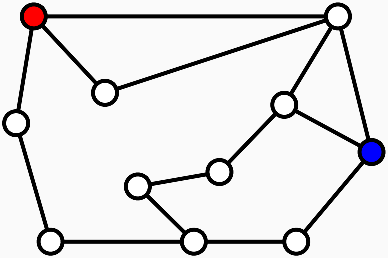
Cuts and Connectivity in Graphs and Hypergraphs

Chao Xu

March 12, 2018

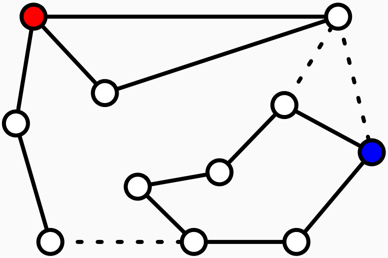
University of Illinois, Urbana-Champaign

Min *st*-cut in graphs



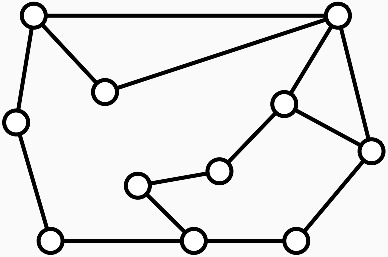
Remove min number of edges to disconnect s from t.

Min st-cut in graphs



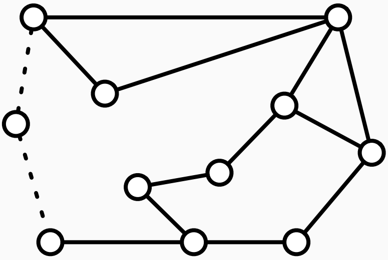
Remove min number of edges to disconnect s from t.

Min cut in graphs



Remove min number of edges to disconnect some pair of vertices.

Min cut in graphs



Remove min number of edges to disconnect some pair of vertices.

Formal definition

Problem: Min *st*-cut

Input: $G = (V, E)$ and $s, t \in V$

Output: A 2-partition (S, T) of the vertices V, such that $s \in S$, $t \in T$, and the number of edges crossing S is minimized.

Problem: Min-cut

Input: $G = (V, E)$

Output: A 2-partition (S, T) of the vertices V, such that the number of edges crossing S is minimized.

Min *st*-cut problem is the fixed-terminal variant of min-cut problem, the global variant.

Formal definition

Problem: Min *st*-cut

Input: $G = (V, E)$ and $s, t \in V$

Output: A 2-partition (S, T) of the vertices V, such that $s \in S$, $t \in T$, and the number of edges crossing S is minimized.

Problem: Min-cut

Input: $G = (V, E)$

Output: A 2-partition (S, T) of the vertices V, such that the number of edges crossing S is minimized.

Min *st*-cut problem is the fixed-terminal variant of min-cut problem, the global variant. The value of the min-cut is also called the (edge) connectivity, denoted λ.

Applications of min-cut and min-st-cut

- Disconnect railroad networks,
- Maximum cardinality bipartite matching,
- Image segmentation,
- ...

Algorithms for graph min-cut

- Finding a min-cut reduces to finding min-*st*-cut for each pair of s and t.
- $\tilde{O}(mn)$ time: Maximum adjacency ordering. [Stoer-Wagner 95].
- $\tilde{O}(m)$ time (randomized). [Karger 98]
- $\tilde{O}(m)$ time (simple, unweighted). [Kawarabayashi-Thorup 15, Henzinger-Rao-Wang 17]

The Thesis

- Efficient algorithms for min-cut and its generalizations in graphs and hypergraphs.
- Understand the complexity difference between global and fixed-terminal variants.

The Thesis

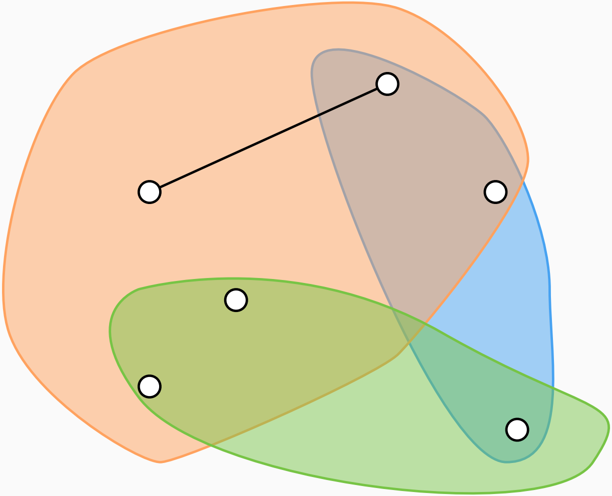
- Efficient algorithms for min-cut and its generalizations in graphs and hypergraphs.
- Understand the complexity difference between global and fixed-terminal variants.

- Algorithms for hypergraph min-cut.
- Approximation for bicut.
- Hypergraph k-cut.
- Minimum violation.

Algorithms for hypergraph min-cut

A hypergraph

A hypergraph $H = (V, E)$ consists of vertices V and edges E.

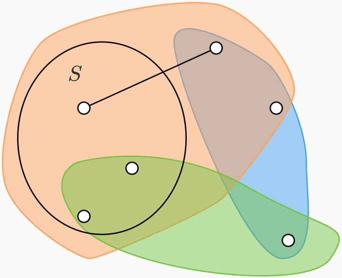


Cut function

- $\delta_H(S)$ is the set of all edges cross S
- The **cut function** $c : 2^V \to \mathbb{N}$

$$c_H(S) = |\delta_H(S)|$$

- A set of vertices $\emptyset \subsetneq S \subsetneq V$ is a min-cut if $c_H(S)$ is minimized.



Motivations for studying hypergraphs

- a natural generalization of graphs
- element connectivity [Chekuri-Rukkanchanunt-X 16]
- real life applications: VLIS, security, data mining, chemistry
  . . .

The min-cut problems

- Find a min-cut.
- Find all min-cuts.
- Find a $(1 + \epsilon)$-approximate min-cut.

Results on hypergraph cuts

	unweighted		weighted	
Problem	graph	hypergraph	graph	hypergraph
Min-cut	$O(m + \lambda n^2)$	$O(p + \lambda n^2)$	$\tilde{O}(mn)$	$\tilde{O}(pn)$
all min-cuts	$O(m + \lambda n^2)$	$O(p + \lambda n^2)$	$\tilde{O}(mn)$	$\tilde{O}(pn)$
$(1+\epsilon)$-min-cut	-	-	$O(m + n^2/\epsilon^2)$	$O(p + n^2 r^4/\epsilon^2)$

- n: # vertices.
- m: # edges.
- p: sum of the cardinality of the edges.
- λ: value of a min-cut.
- r: the rank.

Algorithms for hypergraph min-cut

Min-cut in unweighted hypergraphs

Min-cut in unweighted graphs

1. Find a sparse subgraph with $O(\lambda n)$ edges that preserves the min-cut in $O(m)$ time.
2. Apply the $O(mn)$ min-cut algorithm on the sparse subgraph.

Total running time: $O(m + \lambda n^2)$

k-certificate

Subgraph H' a k-**certificate** of H if for all $S \subseteq V$

$$c_{H'}(S) \geq \min(c_H(S), k).$$

k-certificate

Subgraph H' a k-**certificate** of H if for all $S \subseteq V$

$$c_{H'}(S) \geq \min(c_H(S), k).$$

Example: a spanning tree is a 1-certificate of a connected graph.

k-certificate

Subgraph H' a k-**certificate** of H if for all $S \subseteq V$

$$c_{H'}(S) \geq \min(c_H(S), k).$$

Example: a spanning tree is a 1-certificate of a connected graph.

Theorem ([Nagamochi-Ibaraki 92])
A graph has a k-certificate with $O(kn)$ edges, and can be found in $O(m)$ time.

Application of k-certificate

- finding the connectivity.

Application of k-certificate

- finding the connectivity.
- spanning tree packing. [Gabow 98]

Application of k-certificate

- finding the connectivity.
- spanning tree packing. [Gabow 98]
- sketching in dynamic graph streams. [Guha-McGregor-Tench 15]

Application of k-certificate

- finding the connectivity.
- spanning tree packing. [Gabow 98]
- sketching in dynamic graph streams. [Guha-McGregor-Tench 15]
- $(1+\epsilon)$-approximate min-cut.

k-certificate for hypergraphs?

1. Find a λ-certificate.
2. Apply the $O(pn)$ min-cut algorithm on the λ-certificate.

k-certificate for hypergraphs?

1. Find a λ-certificate.
2. Apply the $O(pn)$ min-cut algorithm on the λ-certificate.

Theorem ([Guha-McGregor-Tench 15])
Every hypergraph has a k-certificate with $O(kn)$ edges(each edge can contain $\Omega(n)$ vertices) and can be found in $O(kp)$ time.

k-certificate for hypergraphs?

1. Find a λ-certificate.
2. Apply the $O(pn)$ min-cut algorithm on the λ-certificate.

Theorem ([Guha-McGregor-Tench 15])
Every hypergraph has a k-certificate with $O(kn)$ edges(each edge can contain $\Omega(n)$ vertices) and can be found in $O(kp)$ time.

Running time $O(\lambda p + \lambda n^3)$.

Small size k-certificate

Theorem ([Chekuri-X 17])
Every hypergraph has a k-certificate with size $O(kn)$, and can be found in $O(p)$ time.

Small size k-certificate

Theorem ([Chekuri-X 17])
Every hypergraph has a k-certificate with size $O(kn)$, and can be found in $O(p)$ time.

Consequence: $O(p + \lambda n^2)$ time algorithm for finding a min-cut.

Algorithms for hypergraph min-cut

All min-cuts

What does it mean to find all min-cuts?

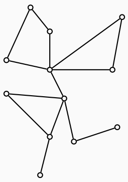
Find a small ($O(n)$-size) data structure that can quickly enumerate/count the number of min-cuts.

A representation

A graph(hypergraph) $G' = (V', E')$ is a representation of $G = (V, E)$, if there exists a function $\phi : V \to V'$ such that

1. S' is a min-cut in G' iff $\phi^{-1}(S')$ is a min-cut in G.
2. S is a min-cut in G, iff $\phi(S)$ is a min-cut in G'.

Graphs: Finding all min-cuts

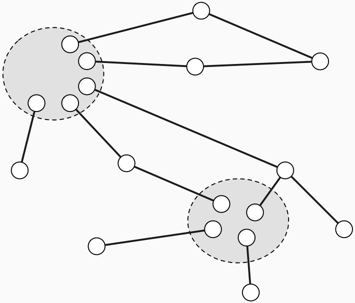


A cactus is a graph in which any two cycles are edge disjoint.

Theorem ([Dinitz et. al. 76, Karzanov & Timofeev 86])
For each graph G there exist a representation G' where G' is a cactus.

A cactus representation can be found

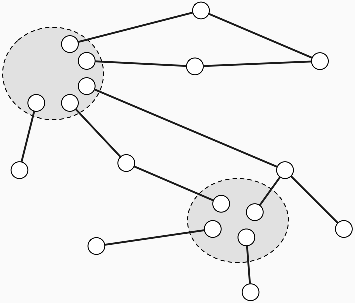
- in $\tilde{O}(nm)$ time [Nagamochi et. al. 03]
- in (randomized) $\tilde{O}(m)$ time [Karger & Panigrahi 09]



Theorem ([Cheng 99, Fleiner & Jordan 99])

For each hypergraph H there exist a representation H' where H' is a hypercactus.

Hypergraphs: Finding all min-cuts



Theorem ([Cheng 99, Fleiner & Jordan 99])

For each hypergraph H there exist a representation H' where H' is a hypercactus.

Expensive to construct, in the order of $O(n^4 p)$.

Hypergraphs: Finding all min-cuts

Theorem ([Chekuri & X 17])

A hypercactus representation can be found in $\tilde{O}(np)$ time.

Hypergraphs: Finding all min-cuts

Theorem ([Chekuri & X 17])

A hypercactus representation can be found in $\tilde{O}(np)$ time.

Approach: Using the decomposition framework [Cunningham 93].

Hypergraphs: Finding all min-cuts

Theorem ([Chekuri & X 17])

A hypercactus representation can be found in $\tilde{O}(np)$ *time.*

Approach: Using the decomposition framework [Cunningham 93].

Matches the result in graphs. Conceptually simpler algorithm.

Algorithms for hypergraph min-cut

$(1+\epsilon)$-approximate min-cut

Finding an $(1+\epsilon)$-approximate min-cut

S is a $(1+\epsilon)$-approximate min-cut if $c(S) \leq (1+\epsilon)\lambda$.

Finding an $(1+\epsilon)$-approximate min-cut

S is a $(1+\epsilon)$-approximate min-cut if $c(S) \leq (1+\epsilon)\lambda$.

1. Find a sparse subgraph that approximately preserves all min-cuts.
2. Find a min-cut in the sparse subgraph.

Cut-sparsifiers

A graph G is a $(1 \pm \epsilon)$-**cut-sparsifier** of H if
$(1-\epsilon)c_G(A) \leq c_H(A) \leq (1+\epsilon)c_G(A)$ for all $A \subseteq V$.

Cut-sparsifiers

A graph G is a $(1 \pm \epsilon)$-**cut-sparsifier** of H if $(1-\epsilon)c_G(A) \leq c_H(A) \leq (1+\epsilon)c_G(A)$ for all $A \subseteq V$.

Theorem ([Benczúr-Karger 98])

There exists a $(1 \pm \epsilon)$-cut-sparsifier of $\tilde{O}(\frac{n}{\epsilon^2})$ edges, and can be constructed in $\tilde{O}(m)$ time with high probability.

Cut-sparsifiers

A graph G is a $(1 \pm \epsilon)$-**cut-sparsifier** of H if $(1-\epsilon)c_G(A) \leq c_H(A) \leq (1+\epsilon)c_G(A)$ for all $A \subseteq V$.

Theorem ([Benczúr-Karger 98])

There exists a $(1 \pm \epsilon)$-cut-sparsifier of $\tilde{O}(\frac{n}{\epsilon^2})$ edges, and can be constructed in $\tilde{O}(m)$ time with high probability.

Consequence: A $(1+\epsilon)$-approximate min-cut can be found in $\tilde{O}(m + \frac{n^2}{\epsilon^2})$ time.

Cut-sparsifiers

A graph G is a $(1 \pm \epsilon)$-**cut-sparsifier** of H if $(1-\epsilon)c_G(A) \le c_H(A) \le (1+\epsilon)c_G(A)$ for all $A \subseteq V$.

Theorem ([Benczúr-Karger 98])

There exists a $(1 \pm \epsilon)$-cut-sparsifier of $\tilde{O}(\frac{n}{\epsilon^2})$ edges, and can be constructed in $\tilde{O}(m)$ time with high probability.

Consequence: A $(1+\epsilon)$-approximate min-cut can be found in $\tilde{O}(m + \frac{n^2}{\epsilon^2})$ time. Near-linear time when graph is dense.

Cut-sparsifiers in hypergraphs

Theorem ([Kogan-Krauthgamer 14])

There exists a $(1 \pm \epsilon)$-cut-sparsifier of $\tilde{O}(\frac{nr}{\epsilon^2})$ edges, and can be constructed in $\tilde{O}(n^2 p)$ time with high probability.

Theorem ([Kogan-Krauthgamer 14])
There exists a $(1 \pm \epsilon)$-cut-sparsifier of $\tilde{O}(\frac{nr}{\epsilon^2})$ edges, and can be constructed in $\tilde{O}(n^2 p)$ time with high probability.

Too slow.

Fast cut-sparsifiers in hypergraphs

Theorem ([Chekuri-X 17])

A $(1 \pm \epsilon)$-cut-sparsifier of H with $\tilde{O}(nr^2/\epsilon^2)$ edges can be found in $\tilde{O}(p)$ time with high probability.

Fast cut-sparsifiers in hypergraphs

Theorem ([Chekuri-X 17])

A $(1 \pm \epsilon)$-cut-sparsifier of H with $\tilde{O}(nr^2/\epsilon^2)$ edges can be found in $\tilde{O}(p)$ time with high probability.

Consequence: $\tilde{O}(p + n^2 r^4/\epsilon^2)$ algorithm for $(1 + \epsilon)$-approximate min-cut in hypergraphs.

Fast cut-sparsifiers in hypergraphs

Theorem ([Chekuri-X 17])

A $(1 \pm \epsilon)$-cut-sparsifier of H with $\tilde{O}(nr^2/\epsilon^2)$ edges can be found in $\tilde{O}(p)$ time with high probability.

Consequence: $\tilde{O}(p + n^2 r^4/\epsilon^2)$ algorithm for $(1 + \epsilon)$-approximate min-cut in hypergraphs. $\tilde{O}(p + n^2/\epsilon^2)$ for constant rank hypergraphs.

Fast cut-sparsifiers in hypergraphs

Theorem ([Chekuri-X 17])

A $(1 \pm \epsilon)$-cut-sparsifier of H with $\tilde{O}(nr^2/\epsilon^2)$ edges can be found in $\tilde{O}(p)$ time with high probability.

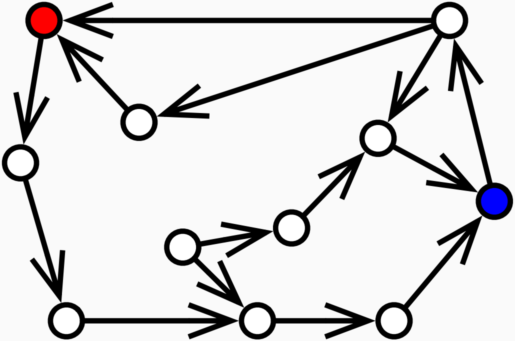
Consequence: $\tilde{O}(p + n^2 r^4/\epsilon^2)$ algorithm for $(1+\epsilon)$-approximate min-cut in hypergraphs. $\tilde{O}(p + n^2/\epsilon^2)$ for constant rank hypergraphs.

Near-linear time when the hypergraph is dense.

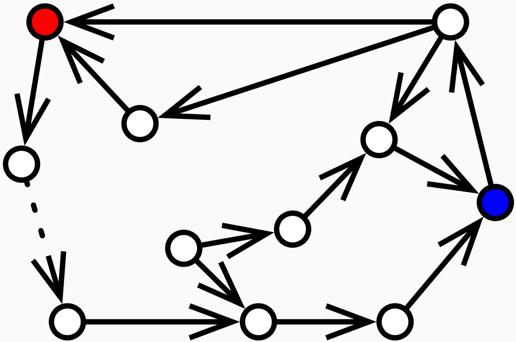
Summary: Fast hypergraph cut algorithms that match their state-of-the-art graph counterparts.

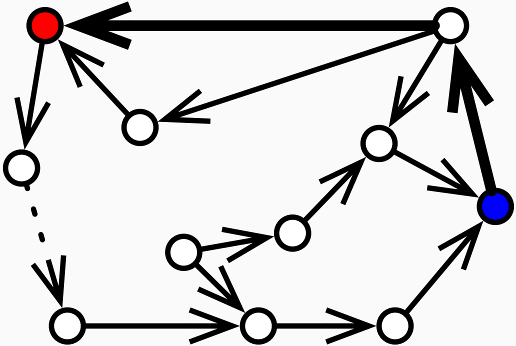
Min-cut in directed graphs: Bicut

st-cut

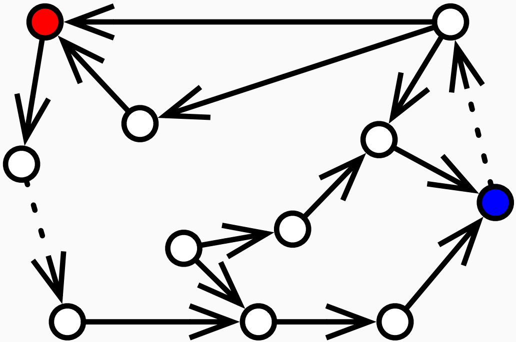


st-cut





st-bicut



Bicut: Generalization of min-cut in directed graphs

- *st*-bicut: A set of edges such that its removal disconnect s and t in both direction.
- bicut: A *st*-bicut for some s and t.

Hardness: min-st-bicut

A special case of multicut in directed graphs.

- Trivial 2-approximation. Union of min-st-cut and min-ts-cut [Dahlhaus et. al. 1994].
- $(2-\epsilon)$-inapproximable under UGC. [Chekuri & Madan 16, Lee 16]

Min-Bicut

It is not known if computing bicut is NP-hard.

Min-Bicut

It is not known if computing bicut is NP-hard.

Theorem ([Bérczi-Chandrasekaran-Király-Lee-X 17])

A $(2-\delta)$-approximation exists for min-bicut, where $\delta = \frac{1}{448}$.

Min-Bicut

It is not known if computing bicut is NP-hard.

Theorem ([Bérczi-Chandrasekaran-Király-Lee-X 17])

A $(2-\delta)$-approximation exists for min-bicut, where $\delta = \frac{1}{448}$.

A hardness separation between fixed-terminal and global bicut!

Vertex based interpretation of 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 two uncomparable sets $A, B \subseteq V$ with minimum $|\delta^{in}(A) \cup \delta^{in}(B)|$.

Vertex based interpretation of 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 two uncomparable sets $A, B \subseteq V$ with minimum $|\delta^{in}(A) \cup \delta^{in}(B)|$.

Approach: Find multiple relaxations such that one of them is a $(2 - \delta)$-approximation.

Vertex based interpretation of 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 two uncomparable sets $A, B \subseteq V$ with minimum $|\delta^{in}(A) \cup \delta^{in}(B)|$.

Approach: Find multiple relaxations such that one of them is a $(2-\delta)$-approximation.
Example: Find uncomparable sets A and B such that $|\delta^{in}(A)|+|\delta^{in}(B)|$ is minimized. If it is not a $(2-\delta)$-approximation, then most edges in the optimal bi-cut are going into $A \cap B$.

Summary: A hardness gap between global and fixed-terminal bicut.

k-cut in hypergraphs

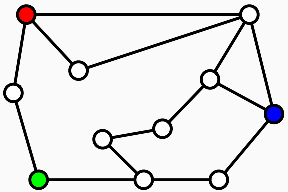
k-way-cut in graphs

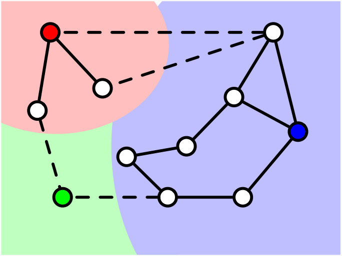
Problem: Min k-way cut

Input: G and $v_1, \ldots, v_k \in V(G)$

Output: A k-partition $(V_1, \ldots, V_k)$, such that $v_i \in V_i$ for all i, and the number of edges crossing the partition classes is minimized.

A min-k-cut is the minimum over all k-way-cut.





Global vs. Fixed-terminal

- Min k-way-cut is hard for $k \geq 3$. [Dahlhaus et. al. 94]

Global vs. Fixed-terminal

- Min k-way-cut is hard for $k \geq 3$. [Dahlhaus et. al. 94]
- Min k-cut. Multiple polynomial time algorithms!
    - Fix a partition class: $n^{\Theta(k^2)}$ [Goldschmidt-Hochbaum 94].

Global vs. Fixed-terminal

- Min k-way-cut is hard for $k \geq 3$. [Dahlhaus et. al. 94]
- Min k-cut. Multiple polynomial time algorithms!
  - Fix a partition class: $n^{\Theta(k^2)}$ [Goldschmidt-Hochbaum 94].
  - Randomized contraction: $\tilde{O}(n^{2(k-1)})$ [Karger-Stein 96].

Global vs. Fixed-terminal

- Min k-way-cut is hard for $k \geq 3$. [Dahlhaus et. al. 94]
- Min k-cut. Multiple polynomial time algorithms!
  - Fix a partition class: $n^{\Theta(k^2)}$ [Goldschmidt-Hochbaum 94].
  - Randomized contraction: $\tilde{O}(n^{2(k-1)})$ [Karger-Stein 96].
  - Divide and conquer: $O(n^{(4+o(1))k})$ [Kamidoi-Yoshida-Nagamochi 07].
  - Divide and conquer: $O(n^{(4-o(1))k})$ [Xiao 08].

Global vs. Fixed-terminal

- Min k-way-cut is hard for $k \geq 3$. [Dahlhaus et. al. 94]
- Min k-cut. Multiple polynomial time algorithms!
  - Fix a partition class: $n^{\Theta(k^2)}$ [Goldschmidt-Hochbaum 94].
  - Randomized contraction: $\tilde{O}(n^{2(k-1)})$ [Karger-Stein 96].
  - Divide and conquer: $O(n^{(4+o(1))k})$ [Kamidoi-Yoshida-Nagamochi 07].
  - Divide and conquer: $O(n^{(4-o(1))k})$ [Xiao 08].
  - Tree packing: $\tilde{O}(n^{2k})$ [Thorup 08].

What about hypergraphs?

Previous works on HYPERGRAPH k-cut

- Min k-way-cut is hard for $k \geq 3$.
- Min k-cut.
  - $k = 2$: Hypergraph cut.

Previous works on HYPERGRAPH k-cut

- Min k-way-cut is hard for $k \geq 3$.
- Min k-cut.
  - $k = 2$: Hypergraph cut.
  - $k = 3$: Deterministic contraction [Xiao 08].

Previous works on HYPERGRAPH k-cut

- Min k-way-cut is hard for $k \geq 3$.
- Min k-cut.
  - $k = 2$: Hypergraph cut.
  - $k = 3$: Deterministic contraction [Xiao 08].
  - Constant rank: Hypertree packing [Fukunaga 10].

Previous works on HYPERGRAPH k-cut

- Min k-way-cut is hard for $k \geq 3$.
- Min k-cut.
  - $k = 2$: Hypergraph cut.
  - $k = 3$: Deterministic contraction [Xiao 08].
  - Constant rank: Hypertree packing [Fukunaga 10].

Main question: Hypergraph k-cut for $k \geq 4$ in arbitrary rank hypergraphs?

Fixed-terminal vs. global complexity gap?

Our result

Theorem ([Chandrasekaran-X-Yu 18])
There exists a randomized polynomial time algorithm that finds a minimum k-cut in a hypergraph.

Our result

Theorem ([Chandrasekaran-X-Yu 18])
There exists a randomized polynomial time algorithm that finds a minimum k-cut in a hypergraph.

Approach:
Randomized contraction algorithm with dampened sampling.

Our result

Theorem ([Chandrasekaran-X-Yu 18])
There exists a randomized polynomial time algorithm that finds a minimum k-cut in a hypergraph.

Approach:
Randomized contraction algorithm with dampened sampling.

Theorem ([Chandrasekaran-X-Yu 18])
There are $O(n^{2(k-1)})$ distinct min-k-cuts in a hypergraph.

Summary: There is a global vs. fixed-terminal complexity gap for hypergraph k-cut.

Minimum violation

Violation

A map from $G = (V, E)$ to $H = (U, F)$ is a function $f : V \to U$.

H is the pattern graph.

An edge $uv \in E$ is a violating edge, if $f(u)f(v) \notin F$.

The violation of f is the number of violating edges.

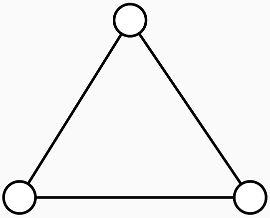
Violation

A map from $G = (V, E)$ to $H = (U, F)$ is a function $f : V \to U$.

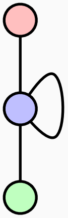
H is the pattern graph.

An edge $uv \in E$ is a violating edge, if $f(u)f(v) \notin F$.

The violation of f is the number of violating edges.



G



H

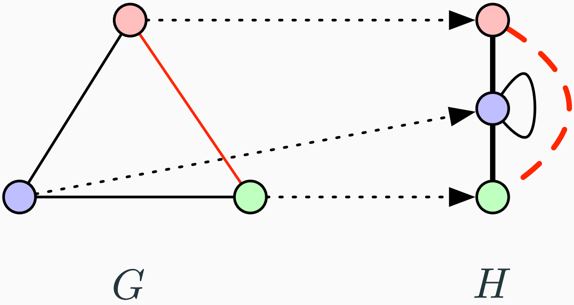
Violation

A map from $G = (V, E)$ to $H = (U, F)$ is a function $f : V \to U$.

H is the pattern graph.

An edge $uv \in E$ is a violating edge, if $f(u)f(v) \notin F$.

The violation of f is the number of violating edges.

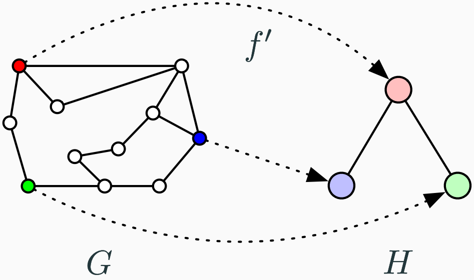


Minimum violation retraction. RVio(H)

Input: graph G and a bijection $f' : V' \to U$ for some $V' \subseteq V(G)$

Output: A map f from G to H such that $f|_{V'} = f'$ and the violation is minimized.

Vertices in V' are fixed vertices.

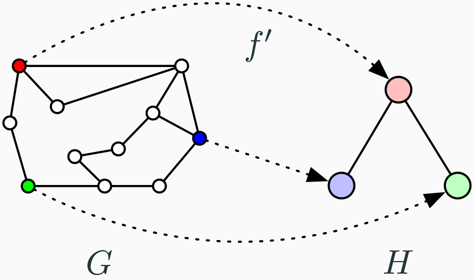


Minimum violation retraction. RVio(H)

Input: graph G and a bijection $f' : V' \to U$ for some $V' \subseteq V(G)$

Output: A map f from G to H such that $f|_{V'} = f'$ and the violation is minimized.

Vertices in V' are fixed vertices.

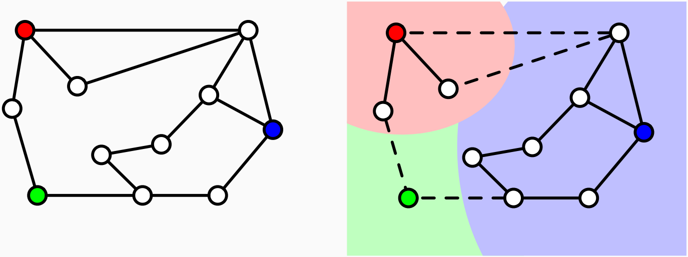


H is r-tractable if **RVio**(H) is tractable.

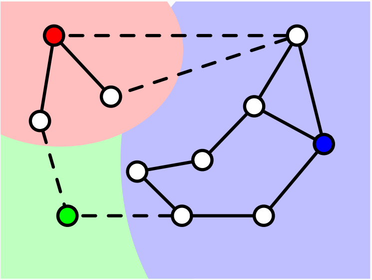
Problem: Min k-way cut

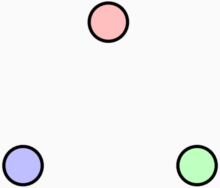
Input: G and $v_1, \ldots, v_k \in V(G)$

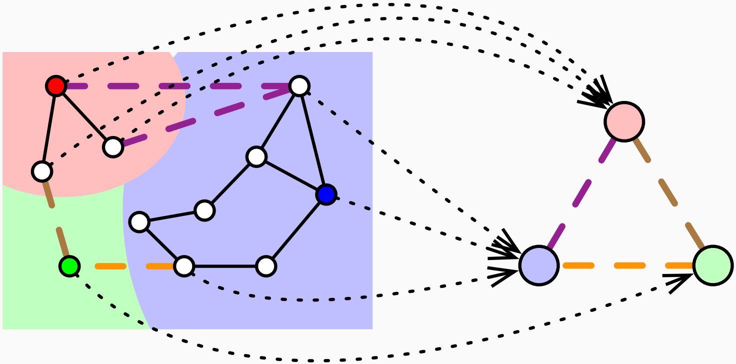
Output: A k-partition $(V_1, \ldots, V_k)$, such that $v_i \in V_i$ for all i, and the number of edges crossing the partition classes is minimized.



3-way cut







Surjective Minimum Violation. SVio(H)

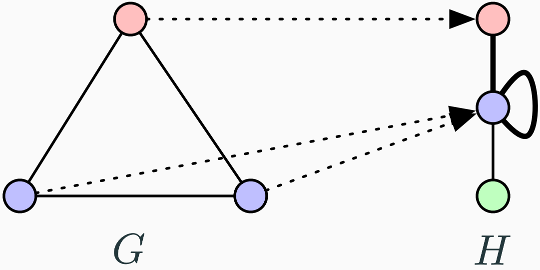
Input: $G = (V, E)$.

Output: A surjective map from G to H with minimum violation.

Surjective Minimum Violation. SVio(H)

Input: $G = (V, E)$.

Output: A surjective map from G to H with minimum violation.

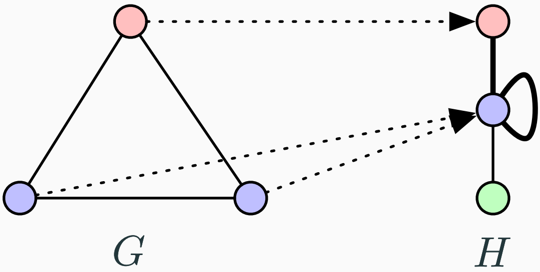


NOT SURJECTIVE!

Surjective Minimum Violation. SVio(H)

Input: $G = (V, E)$.

Output: A surjective map from G to H with minimum violation.



NOT SURJECTIVE! H is s-tractable if **SVio**(H) is tractable.

Why minimum violation?

Complete classification of r-tractable/s-tractable graphs implies complexity of various cut problems.

Why minimum violation?

Complete classification of r-tractable/s-tractable graphs implies complexity of various cut problems.

Classification of s-tractable graphs and r-tractable graphs was studied under the name "G_c-cut". [Elem-Hassin-Monnot 13]

Goal: classify the s-tractable and r-tractable graphs.

Classification of r-tractable (directed) graphs

Theorem ([Kawarabayashi-X unpublished])
There exists a polynomial time algorithm that decides if a (directed) graph is r-tractable.

Classification of r-tractable graphs

v dominates u if $N(u) \subsetneq N(v)$.

A graph $G = (V, E)$ is a double-clique, if $G = G[A] \cup G[B]$ for some clique $A, B \subseteq V$.

Theorem ([Kawarabayashi-X unpublished])
A reflexive graph G is r-tractable if and only if $G[U]$ is a double-clique, where U is the set of non-dominated vertices.

A theorem on s-tractable graphs

Theorem ([Kawarabayashi-X unpublished])
A reflexive graph H is s-tractable if and only if each of its component is s-tractable.

A theorem on s-tractable graphs

Theorem ([Kawarabayashi-X unpublished])
A reflexive graph H is s-tractable if and only if each of its component is s-tractable.

Consequences:

- k-cut is solvable in polynomial time.

A theorem on s-tractable graphs

Theorem ([Kawarabayashi-X unpublished])
A reflexive graph H is s-tractable if and only if each of its component is s-tractable.

Consequences:

- k-cut is solvable in polynomial time.
- Size-constrained k-cut: each partition class has at least c (a constant) vertices is solvable in polynomial time.

Thank you!