Minimum violation maps and their applications to cut problems

Ken-ichi Kawarabayashi, **Chao Xu**

Oct 9, 2017

University of Illinois, Urbana-Champaign

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.

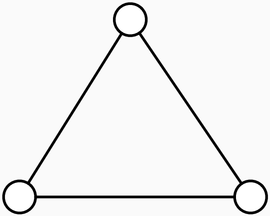
Violation

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

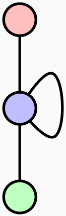
H is the pattern graph.

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

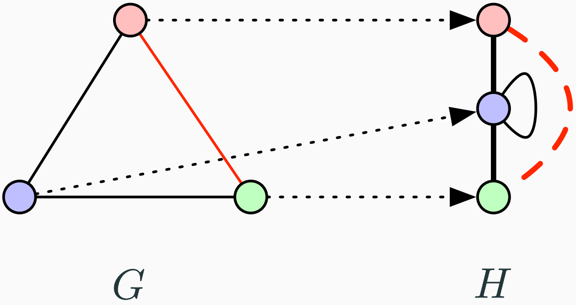
Violation

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. MinVio(H)

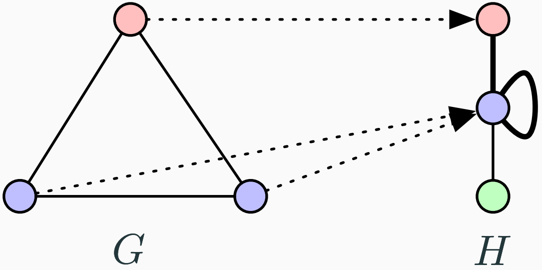
Input: $G = (V, E)$.

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

Minimum Violation. MinVio(H)

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

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

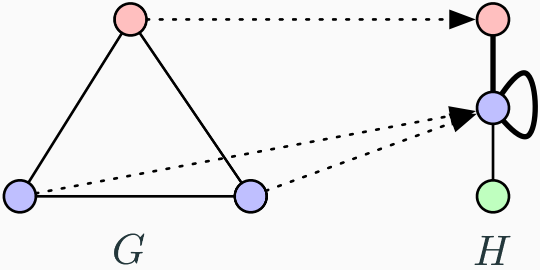


BAD! NOT SURJECTIVE!

Minimum Violation. MinVio(H)

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

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



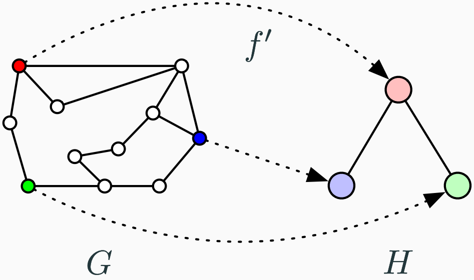
BAD! NOT SURJECTIVE! H is s-tractable if **MinVio**(H) is tractable.

Fixed-terminal minimum violation. FixMinVio(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.

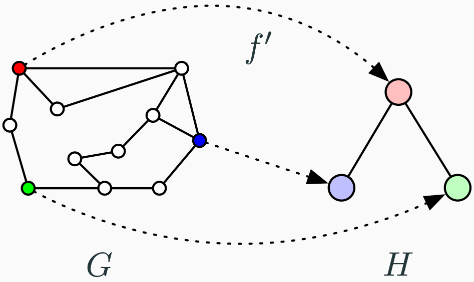


Fixed-terminal minimum violation. FixMinVio(H)

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 f-tractable if **FixMinVio**(H) is tractable.

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

Why minimum violation?

- A homomorphism is a map with violation 0.

Why minimum violation?

- A homomorphism is a map with violation 0.
- **FixMinVio**(H) models fixed terminal cut problems.

Why minimum violation?

- A homomorphism is a map with violation 0.
- **FixMinVio**(H) models fixed terminal cut problems.
- **MinVio**(H) models global cut problems.

Why minimum violation?

- A homomorphism is a map with violation 0.
- **FixMinVio**(H) models fixed terminal cut problems.
- **MinVio**(H) models global cut problems.
- A complete classification of f-tractable/s-tractalbe graphs implies complexity of various cut problems.

Why minimum violation?

- A homomorphism is a map with violation 0.
- **FixMinVio**(H) models fixed terminal cut problems.
- **MinVio**(H) models global cut problems.
- A complete classification of f-tractable/s-tractalbe graphs implies complexity of various cut problems.

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

Outline

- Model cut problems by minimum violation maps.
- A complete classification of f-tractable graphs.
- For a reflexive graph, s-tractability only depend on the s-tractability of its components.

Remarks

- All graphs after this point are reflexive. For simplicity, we do not draw the self-loops.
- We state theorems for graphs, but there are directed graph counterparts.
- Our results hold for weighted graphs too. The violation is the sum of the weights of the violating edges.

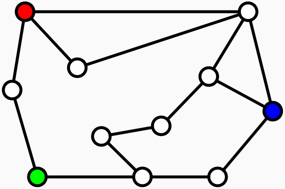
Modeling cut problems

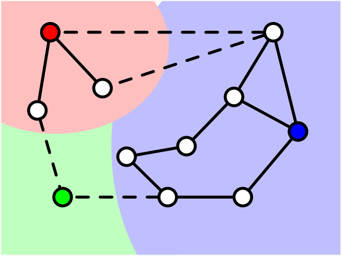
k-way cut

Problem: Min k-way cut

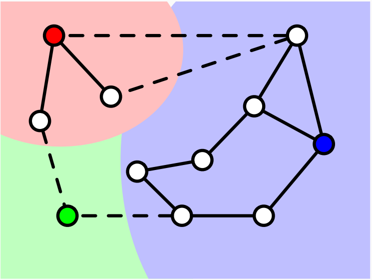
Input: G and $v_1, \ldots, v_k \in V(G)$

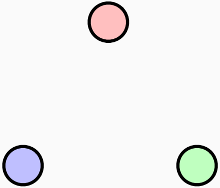
Output: A k-partition $(V_1, \ldots, V_k)$, such that $v_i \in V_i$ for all i, and the number of edges crossing the partition classes is minimized.



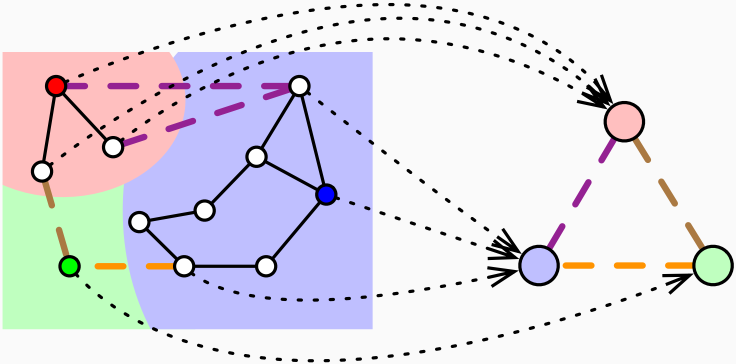


3-way cut





3-way cut



k-cut

Problem: min k-cut

Input: $G = (V, E)$

Output: A k-partition $V_1, \ldots, V_k$ of V, such that the number of edges crossing the partition classes is minimized.

k-way cut and k-cut

T_k is the graph of k isolated vertices, each with a self-loop.

k-way cut is equivalent to **FixMinVio**(T_k).

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

k-cut is equivalent to **MinVio**(T_k).

Solvable in polynomial time for every fixed k **[Goldschmidt & Hochbaum 94, Karger & Stein 96]**.

ℓ-length-bounded cut

ℓ-length bounded *st*-cut is a set of edges that intersect every *st*-path of length at most ℓ.

ℓ-length-bounded cut

ℓ-length bounded *st*-cut is a set of edges that intersect every *st*-path of length at most ℓ.

∞-length bounded *st*-cut is the standard *st*-cut.

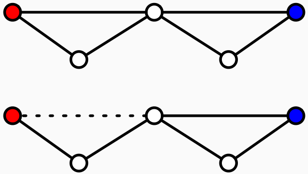
ℓ-length bounded *st*-cut is a set of edges that intersect every *st*-path of length at most ℓ.

∞-length bounded *st*-cut is the standard *st*-cut.

Problem: ℓ-length-bounded cut

Input: G and a pair of vertices s and t

Output: A minimum cardinality ℓ-length-bounded *st*-cut.



2-length bounded cut

ℓ-length-bounded cut

Theorem ([Mahjoub & McCormick 00])
ℓ-length-bounded cut is tractable if and only if $\ell \leq 3$.

ℓ-length-bounded cut

Theorem ([Mahjoub & McCormick 00]**)**

ℓ-length-bounded cut is tractable if and only if $\ell \leq 3$.

Let P_k be a path on k vertices.

Theorem

ℓ-length-bounded cut is equivalent to **FixMinVio**$(P_{\ell+2})$.

min-k-subpartition

A k-subpartition of V is k pairwise disjoint non-empty sets contained in V.

min-k-subpartition

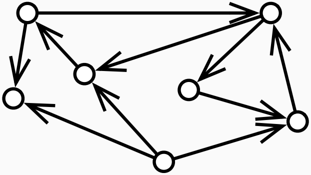
A k-subpartition of V is k pairwise disjoint non-empty sets contained in V.

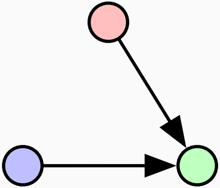
Problem: Min k-subpartition

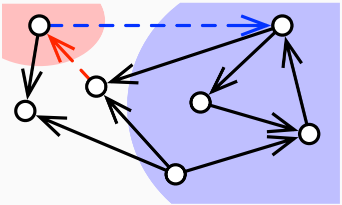
Input: Directed graph $G = (V, E)$

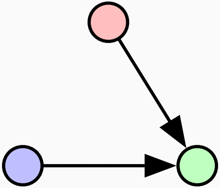
Output: A k-subpartition of V, $\{V_1, \ldots, V_k\}$ where $\sum_{i=1}^{k} |\delta^{in}(V_i)|$ is minimized.

2-subpartition

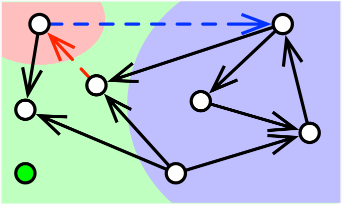


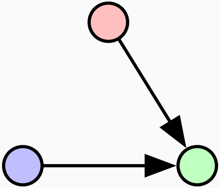


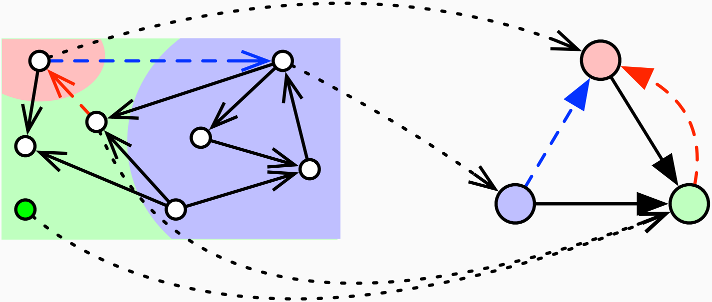




2-subpartition

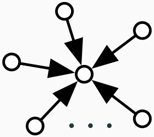




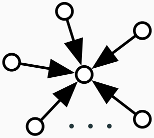


k-subpartition

S_k is a directed star with k leaves and all its edges oriented toward the center.



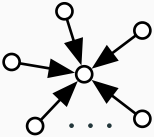
S_k is a directed star with k leaves and all its edges oriented toward the center.



Finding a min-k-subpartition is equivalent to solving **MinVio**(S_k).

k-subpartition

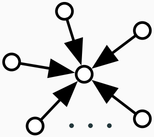
S_k is a directed star with k leaves and all its edges oriented toward the center.



Finding a min-k-subpartition is equivalent to solving **MinVio**(S_k).

Tractable for $k = 2$. Flow based algorithm. **[Bernáth, Pap 15]**

S_k is a directed star with k leaves and all its edges oriented toward the center.



Finding a min-k-subpartition is equivalent to solving **MinVio**(S_k).

Tractable for $k = 2$. Flow based algorithm. **[Bernáth, Pap 15]**

Tractable for constant k. Using subtree hypergraph.

Classification of f-tractable graphs

Start with a harder problem.

Cost

Let $G = (V, E)$, $H = (U, F)$. A cost function $c : V \times U \to \mathbb{N}$ assigns cost $c(v, u)$ to mapping v to u.

The cost of a map f from G to H is

$$\sum_{v \in V} c(v, f(v))$$

Minimum cost and violation

Problem: MinCostVio(H)

Input: Graph G and a cost function c.

Output: A map f from G to H that minimizes the sum of violation and cost.

Minimum cost and violation

Problem: MinCostVio(H)

Input: Graph G and a cost function c.

Output: A map f from G to H that minimizes the sum of violation and cost.

H is c-tractable if **MinCostVio**(H) is tractable.

Minimum cost and violation

Problem: MinCostVio(H)

Input: Graph G and a cost function c.

Output: A map f from G to H that minimizes the sum of violation and cost.

H is c-tractable if **MinCostVio**(H) is tractable.

Theorem ([Deineko et.al. 08])

H is c-tractable if and only if its edge set can be partitioned into two cliques, and the two cliques spans the graph.

But why do we care about MinCostVio(H)?

FixMinVio(H) reduces to **MinCostVio**(H).

But why do we care about **MinCostVio**(H)?

FixMinVio(H) reduces to **MinCostVio**(H).

Fixing vertices using cost.

But why do we care about MinCostVio(H)?

FixMinVio(H) reduces to **MinCostVio**(H).

Fixing vertices using cost.

Input of **FixMinVio**(H) is G and $f' : V' \to U$.

Input to **MinCostVio**(H) is G, c, where $c(v', u) = \infty$ if $v' \in V'$ and $f(v') \neq u$ and 0 everywhere else.

But why do we care about MinCostVio(H)?

FixMinVio(H) reduces to **MinCostVio**(H).

Fixing vertices using cost.

Input of **FixMinVio**(H) is G and $f' : V' \to U$.

Input to **MinCostVio**(H) is G, c, where $c(v', u) = \infty$ if $v' \in V'$ and $f(v') \neq u$ and 0 everywhere else.

Hope: c-tractable and f-tractable are the same?

But why do we care about **MinCostVio**(H)?

FixMinVio(H) reduces to **MinCostVio**(H).

Fixing vertices using cost.

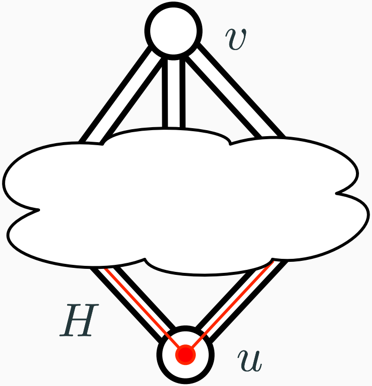
Input of **FixMinVio**(H) is G and $f' : V' \to U$.

Input to **MinCostVio**(H) is G, c, where $c(v', u) = \infty$ if $v' \in V'$ and $f(v') \neq u$ and 0 everywhere else.

Hope: c-tractable and f-tractable are the same?

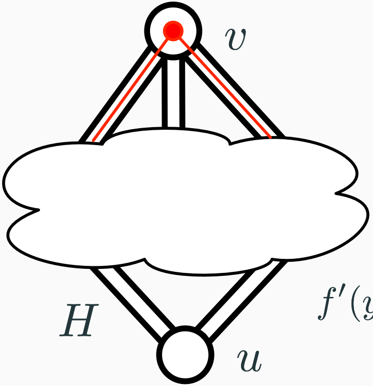
Nope: P_5 is f-tractable but c-tractable.

An observation: moving up



$$N(u) \subseteq N(v)$$
$$f(x) = u$$

An observation: moving up



$$N(u) \subseteq N(v)$$
$$f(x) = u$$

$$f'(y) = \begin{cases} f(y) & \text{if } y \neq x \\ v & \text{otherwise} \end{cases}$$

Violation of f' is at most violation of f.

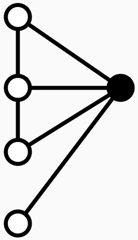
Superseded vertices

Assume there is a total order $\prec$ of the vertices in H. u is superseded by v, if

- $N(u) \subsetneq N(v)$, or
- $N(u) = N(v)$ and $u \prec v$.

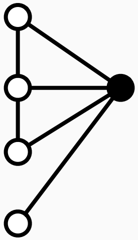
Apex

A vertex is an apex if no vertex supersedes it.



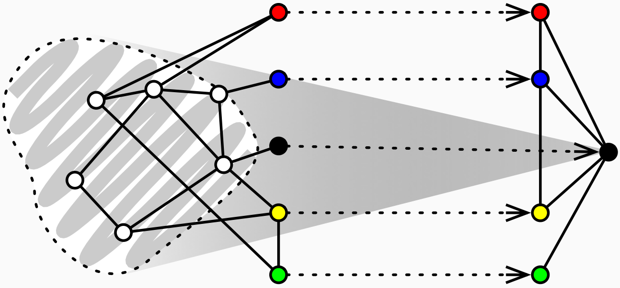
Apex

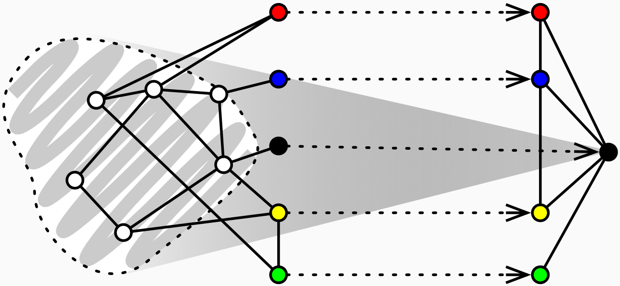
A vertex is an apex if no vertex supersedes it.



The apex subgraph of H is $H[A]$, where A is the set of apex vertices.

There exists an optimal solution where the non-fixed vertices are mapped to apex vertices.





A graph with a single vertex apex subgraph is f-tractable.

f-tractability and c-tractability

Theorem ([Elem, Hassin & Monnot 13])
H is f-tractable if the apex subgraph of H is a complete graph.

Theorem ([Kawarabayashi & X manuscript])

H is f-tractable if the apex subgraph of H is c-tractable.

$$\textbf{FixMinVio}(H)$$

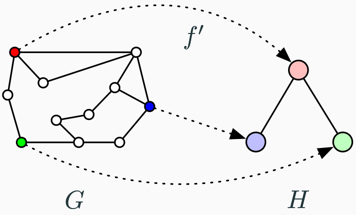
$$G = (V, E)$$

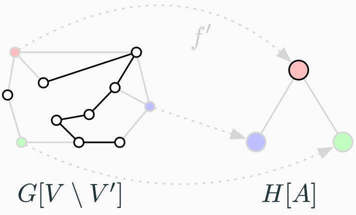
$$f' : V' \to U$$

$$\textbf{MinCostVio}(H[A])$$

$$G[V \setminus V']$$

$$c(v,u) = \left| \left\{ vv' \;\middle|\; \begin{array}{l} v' \in V' \\ vv' \in E \\ uf'(v') \notin F \end{array} \right\} \right|$$





Classification of FixMinVio

Theorem ([Kawarabayashi & X manuscript])
H is f-tractable if and only if the apex subgraph of H is c-tractable.

Theorem ([Kawarabayashi & X manuscript])
H is f-tractable if and only if the apex subgraph of H is c-tractable.

The theorem holds for directed graphs for an appropriate definition of apex.

Consequences

ℓ-length bounded cuts

Theorem ([Mahjoub & McCormick 00])

ℓ-length-bounded cut is tractable if and only if $\ell \leq 3$.

Proof.

ℓ-length-bounded cut is equivalent to **FixMinVio**$(P_{\ell-2})$.

The apex subgraph of P_k is P_{k-2}.

P_k is c-tractable iff $k \leq 3$.

P_k is f-tractable iff $k \leq 5$. $\square$

Consequences

An extremely artificial problem to illustrate a point

Input: Graph G and vertices x, y, z.

Output: A minimum cardinality set of edges F such that

- $d_{G-F}(x, y), d_{G-F}(y, z) \geq 3$,
- $d_{G-F}(x, z) \geq 4$.

Consequences

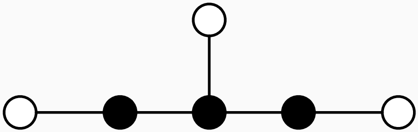
An extremely artificial problem to illustrate a point

Input: Graph G and vertices x, y, z.

Output: A minimum cardinality set of edges F such that

- $d_{G-F}(x, y), d_{G-F}(y, z) \geq 3$,
- $d_{G-F}(x, z) \geq 4$.

Reduces to **FixMinVio**(H), where H is:



Consequences

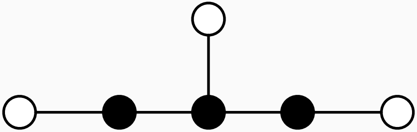
An extremely artificial problem to illustrate a point

Input: Graph G and vertices x, y, z.

Output: A minimum cardinality set of edges F such that

- $d_{G-F}(x, y), d_{G-F}(y, z) \geq 3$,
- $d_{G-F}(x, z) \geq 4$.

Reduces to **FixMinVio**(H), where H is:



Solvable in polynomial time.

s-tractable graphs

We know little about s-tractable graphs

MinVio$_0$(H)

Input: Graph G.

Output: Decide if there is a surjective map from G to H with violation 0.

A graph H is s$_0$-tractable if **MinVio$_0$**(H) is tractable.

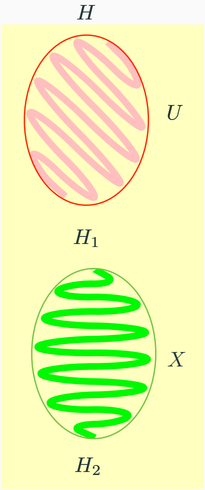
Previously known **[Elem, Hassin & Monnot 13]**:

- H is f-tractable then it is s-tractable.
- H is not s$_0$-tractable then it is not s-tractable.

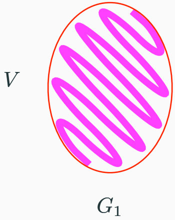
Main Theorem

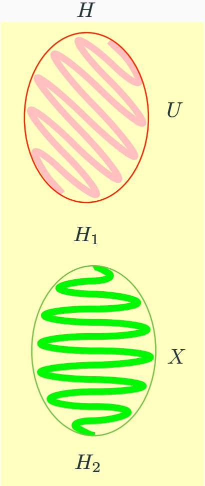
Theorem
A reflexive graph H is s-tractable if and only if each of its component is s-tractable.

Hardness

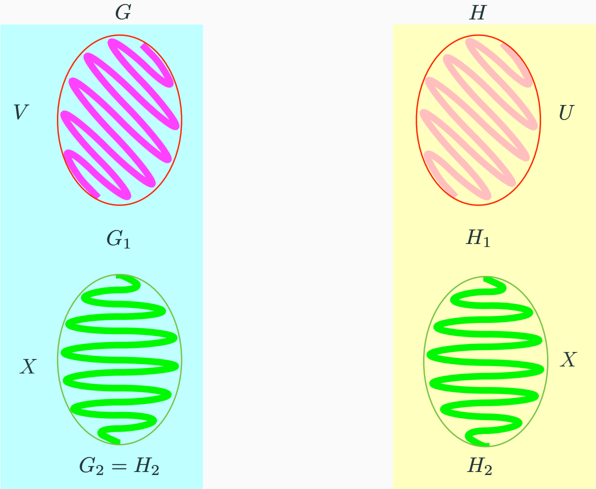


Hardness

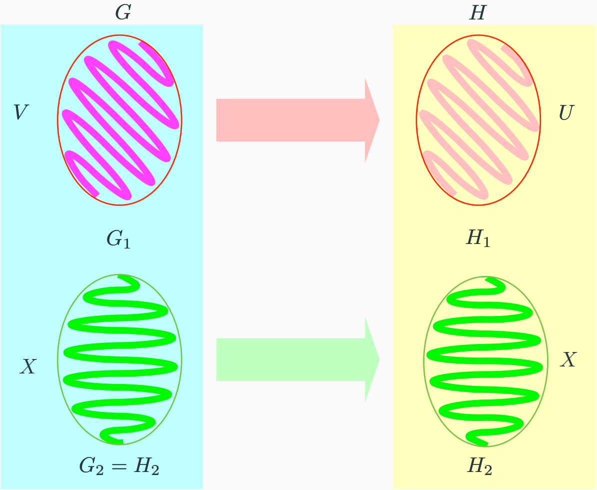




Hardness



Hardness



Hardness

For a minimum surjective map f from G to H, we can find a surjective map f' such that

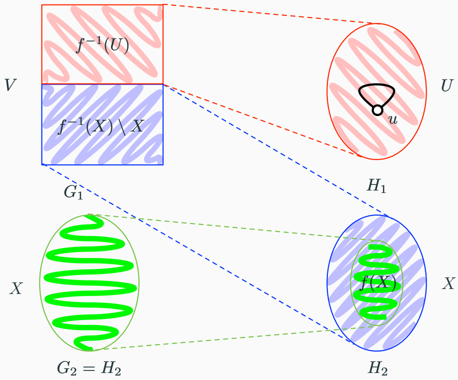
- violation of f' is no larger than violation of f,
- $f'(X) = X$,
- $f'(V) = U$.

$f'|_V$ is the desired minimum violation map from G_1 to H_1.

Consider an optimal solution f.

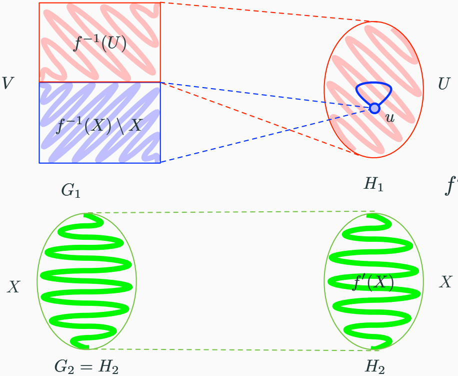
No edge in G_2 is a violating edge.

Hardness



$$f(X) \subset X$$

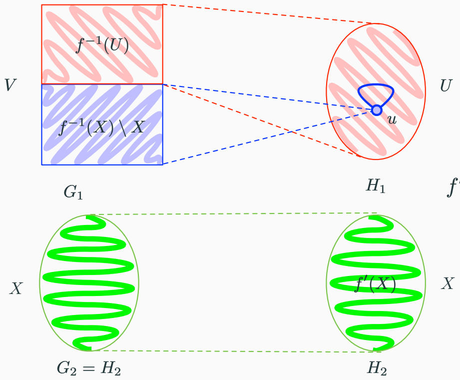
Hardness



$$f(X) \subset X$$

$$f'(v) = \begin{cases} v & \text{if } v \in X \\ u & \text{if } v \in f^{-1}(X) \setminus X \\ f(v) & \text{otherwise} \end{cases}$$

Hardness

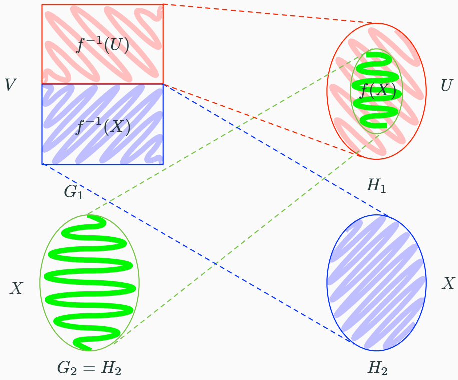


$$f(X) \subset X$$

$$f'(v) = \begin{cases} v & \text{if } v \in X \\ u & \text{if } v \in f^{-1}(X) \setminus X \\ f(v) & \text{otherwise} \end{cases}$$

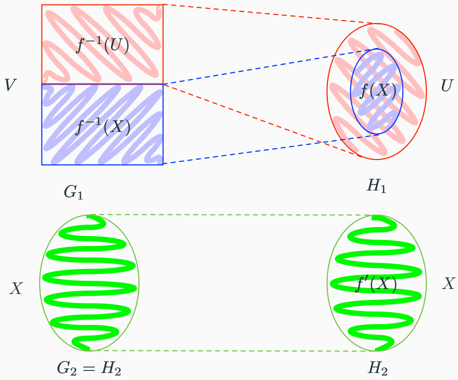
Reflexivity is crucial.

Hardness



$$f(X) \subset U$$

Hardness



$$f(X) \subset U$$

$$f'(v) = \begin{cases} v & \text{if } v \in X \\ f(f(v)) & \text{if } v \in f^{-1}(X) \\ f(v) & \text{otherwise} \end{cases}$$

The other direction: Polynomial time algorithm

Theorem

If the components of a reflexive graph H are s-tractable, then H is s-tractable.

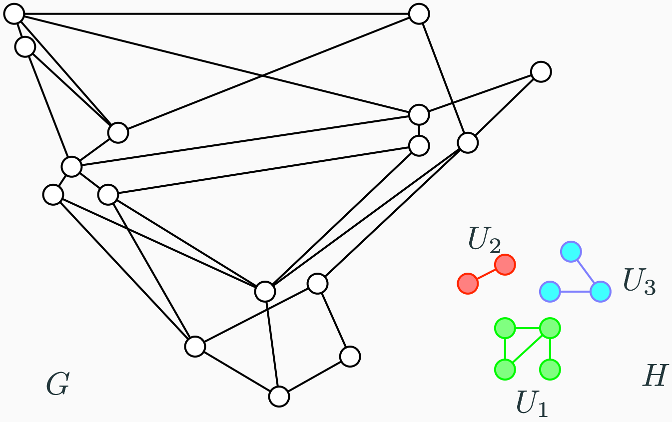
Set up

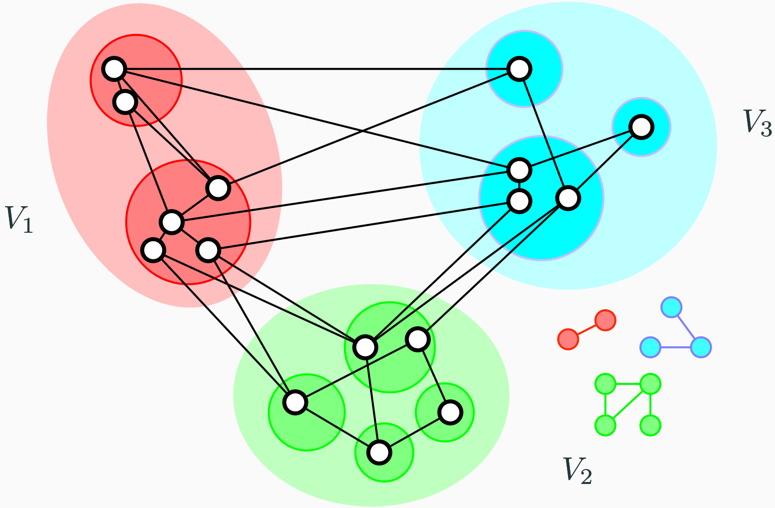
H is a k vertex graph consist of components $U_1, \ldots, U_m$.

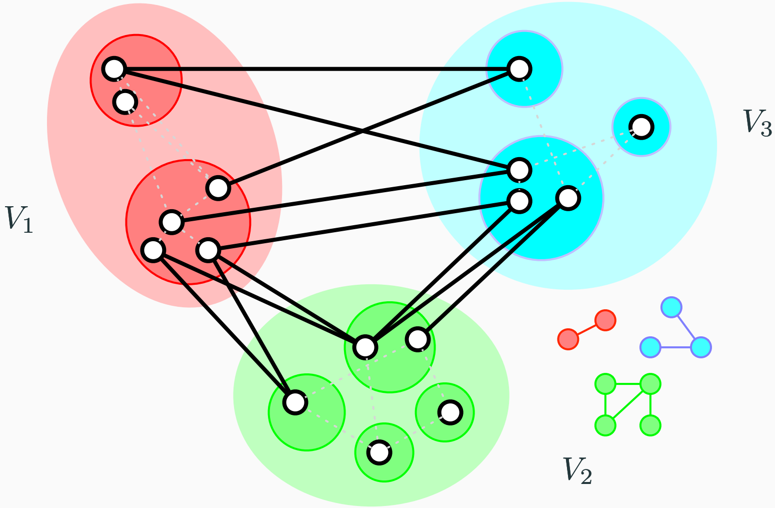
$H[U_i]$ is s-tractable for all i.

f is the optimal solution of **MinVio**(H) with input graph G.

$V_i = f^{-1}(U_i)$.


G
U2
U3
U1
H


V_1
V_3
V_2



V_1
V_3
V_2

The set of edges crossing the m-cut $(V_1, \ldots, V_m)$ has value at most the value of the min-k-cut of G.

min-k-cut value $\geq$ min violation $\geq$ m-cut value.

Greedy Spanning Tree Packing

Theorem ([Thorup 08])
There exists a set of $\tilde{O}(mk^3)$ *spanning trees* $\mathcal{T}$ *such that for each min-k-cut, there is a tree* $T \in \mathcal{T}$ *that crosses it at most* $2(k-1)$ *times.*

Greedy Spanning Tree Packing

Theorem ([Thorup 08])

There exists a set of $\tilde{O}(mk^3)$ spanning trees $\mathcal{T}$ such that for each min-k-cut, there is a tree $T \in \mathcal{T}$ that crosses it at most $2(k-1)$ times.

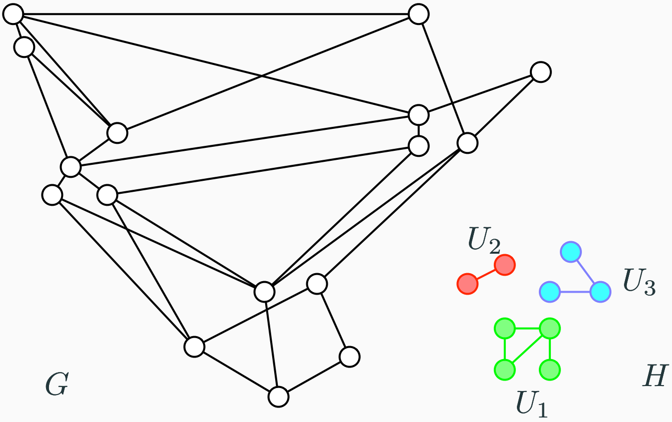
Theorem

There exists a set of $\tilde{O}(mk^3)$ spanning trees $\mathcal{T}$ such that for each set F of edges with weight at most the value of a min-k-cut, there is a tree in $T \in \mathcal{T}$ so $|F \cap T| \leq 2(k-1)$.

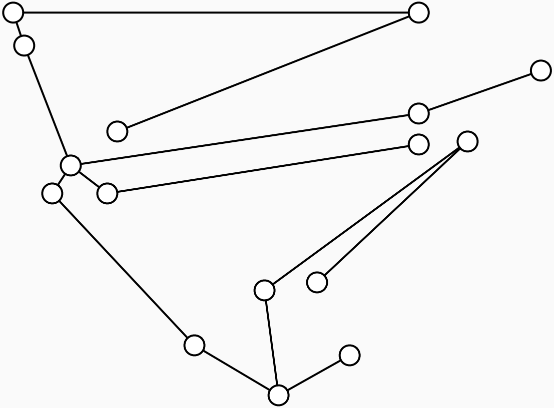
Algorithm

Input graph G.

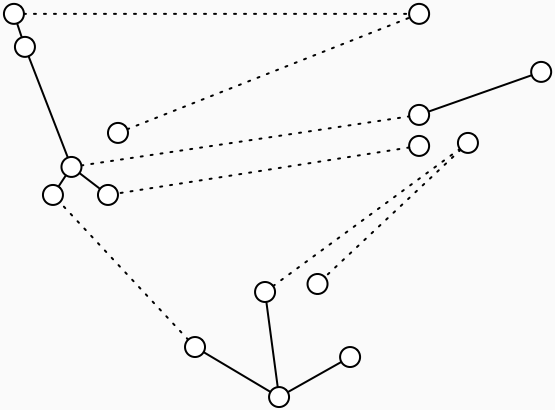
1. Compute the greedy packing of $\tilde{O}(mk^3)$ spanning trees $\mathcal{T}$.
2. For each tree $T \in \mathcal{T}$ and each set of $2(k-1)$ edges F in T:
    2.1 $\mathcal{C} \leftarrow$ the components of $T - F$.
    2.2 For every possible ordered m partition of $\mathcal{C}$ into $(\mathcal{C}_1, \ldots, \mathcal{C}_m)$.
        2.2.1 $V_i \leftarrow \bigcup_{X \in \mathcal{C}_i} X$.
        2.2.2 Solve **MinVio**($H[U_i]$) with input $G[V_i]$.
        2.2.3 Combine the solutions into a candidate solution.
3. Output the minimum candidate solution.



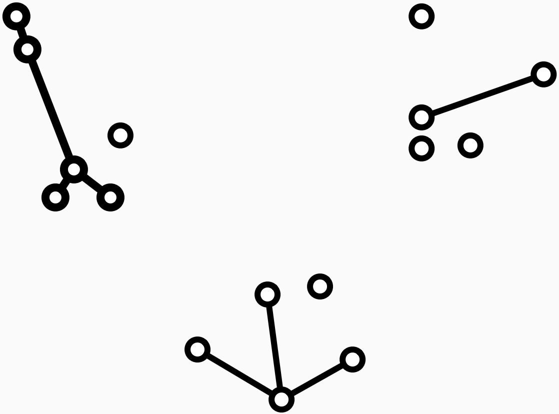
Graph G and H. H consist of components U_1, U_2 and U_3.



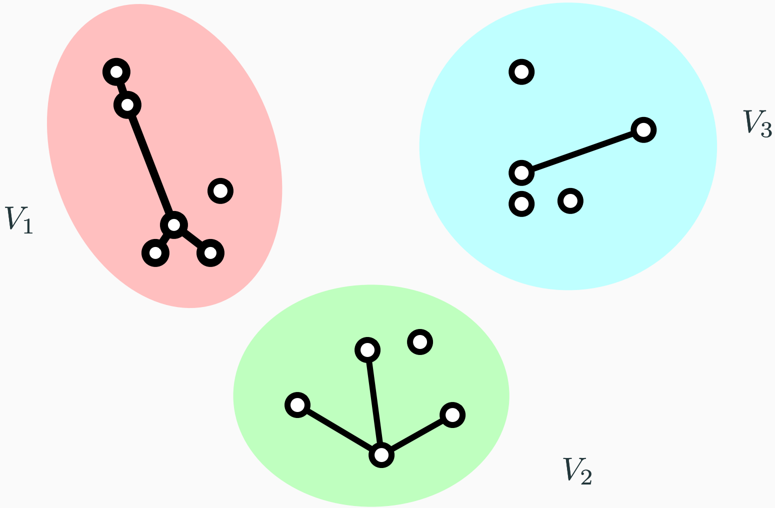
Guess spanning tree T.



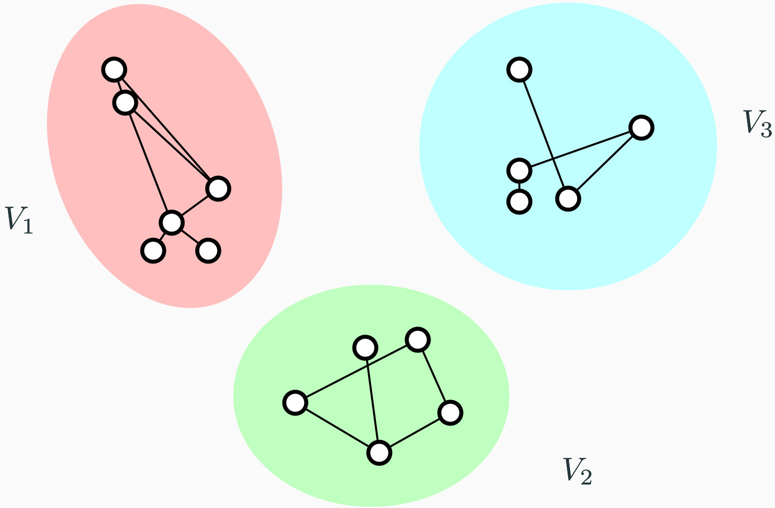
Guess edges F.



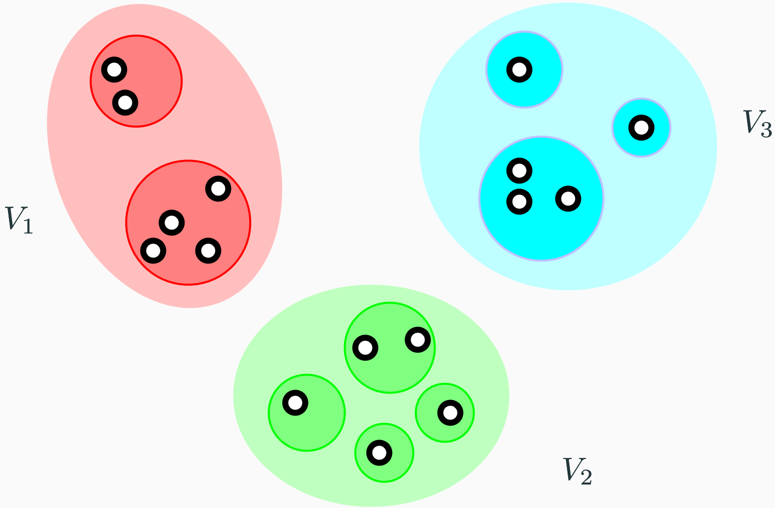
Components of $T - F$.



Guess grouping of the components.



Solve **MinVio**($H[U_i]$) on $G[V_i]$.



The resulting map.

Main Theorem

Theorem
A reflexive graph H is s-tractable if and only if each of its component is s-tractable.

The min s-size-m-cut problem

Let $s = (s_1, \ldots, s_m)$, $s_i \geq s_{i+1} \geq 1$.

min-s-size-m-cut problem

Input: Graph G.

Output: Find a m-partition of the vertices $V_1, \ldots, V_m$ such that $|V_i| \geq s_i$, and total number of edges crossing the partition is minimized.

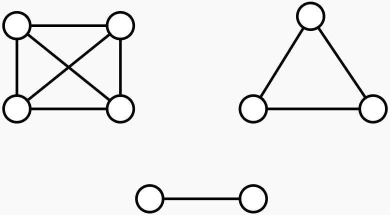
The min s-size-m-cut problem

Let $s = (s_1, \dots, s_m)$, $s_i \geq s_{i+1} \geq 1$.

min-s-size-m-cut problem

Input: Graph G.

Output: Find a m-partition of the vertices $V_1, \dots, V_m$ such that $|V_i| \geq s_i$, and total number of edges crossing the partition is minimized.



Equivalent to **MinVio**$(K_{s_1} \cup \dots K_{s_m})$.

Application to min s-size m-cut

$s = (s_1, \ldots, s_m)$. Let $k = \sum_{i=1}^{m} s_i$.

Application to min s-size m-cut

$s = (s_1, \ldots, s_m)$. Let $k = \sum_{i=1}^{m} s_i$.

- Claimed deterministic $n^{O(k^2)}$ time algorithm. **[Elem, Hassin & Monnot 13]**

Application to min s-size m-cut

$s = (s_1, \ldots, s_m)$. Let $k = \sum_{i=1}^{m} s_i$.

- Claimed deterministic $n^{O(k^2)}$ time algorithm. **[Elem, Hassin & Monnot 13]**
- Randomized $\tilde{O}(n^{2k})$ time algorithm. **[Guiñez & Queyranne 12]**

Application to min s-size m-cut

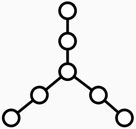
$s = (s_1, \ldots, s_m)$. Let $k = \sum_{i=1}^{m} s_i$.

- Claimed deterministic $n^{O(k^2)}$ time algorithm. **[Elem, Hassin & Monnot 13]**
- Randomized $\tilde{O}(n^{2k})$ time algorithm. **[Guiñez & Queyranne 12]**
- The value of a minimum s-size m-cut is at most the value of min-$(1 + k - s_1)$-cut. (for large enough graphs)
- Deterministic $\tilde{O}(n^{2(k-s_1)})$ time algorithm.

Open Problems

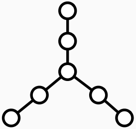
Classify the s-tractable graphs

- Trees?



Classify the s-tractable graphs

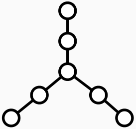
- Trees?



Given a graph G, delete minimum number of edges such that there exists 3 vertices with pairwise distance at least 4.

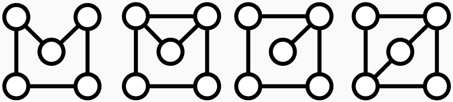
Classify the s-tractable graphs

- Trees?

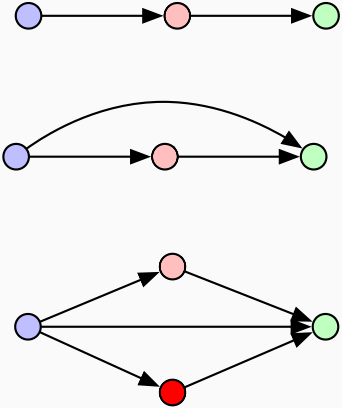


  Given a graph G, delete minimum number of edges such that there exists 3 vertices with pairwise distance at least 4.

- 5 vertex graphs?



Classify the s-tractable directed graphs



Equivalent to global linear-3-cut and global bicut, respectively **[BCKLX 17]**.

Thank you