Minimum violation maps and their applications to cut problems

Ken-ichi Kawarabayashi, Chao Xu

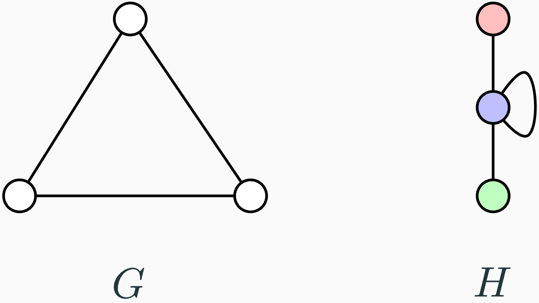
March 4, 2022

University of Electronic Science and Technology of China

Map that preserve graph structure

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

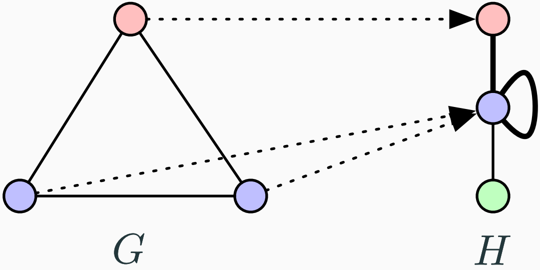
A map is a (graph) homomorphism, if $uv \in E$ then $f(u)f(v) \in F$.



Map that preserve graph structure

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

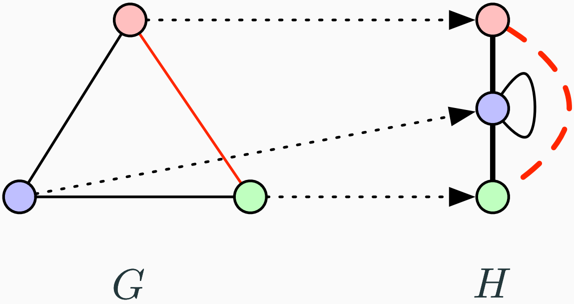
A map is a (graph) homomorphism, if $uv \in E$ then $f(u)f(v) \in F$.



Map that preserve graph structure

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

A map is a (graph) homomorphism, if $uv \in E$ then $f(u)f(v) \in F$.



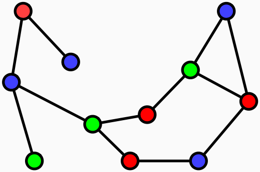
Graph homomorphism model other problems

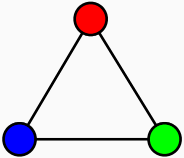
Fix a pattern graph, is the graph homomorphism problem hard?

Graph homomorphism model other problems

Fix a pattern graph, is the graph homomorphism problem hard?

Is the graph 3-colorable?

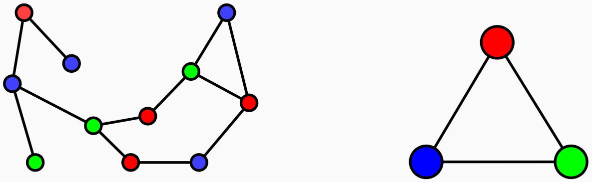




Graph homomorphism model other problems

Fix a pattern graph, is the graph homomorphism problem hard?

Is the graph 3-colorable?



Graph homomorphism is NP-hard.

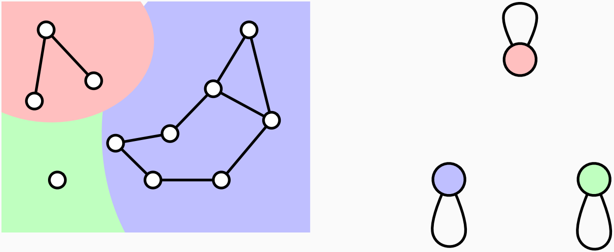
Graph homomorphism can be easy (even with constratints)

The graph homomorphism problem for any graph with a self-loop is trivial: map all vertices to a vertex with self loop.

Graph homomorphism can be easy (even with constratints)

The graph homomorphism problem for any graph with a self-loop is trivial: map all vertices to a vertex with self loop.

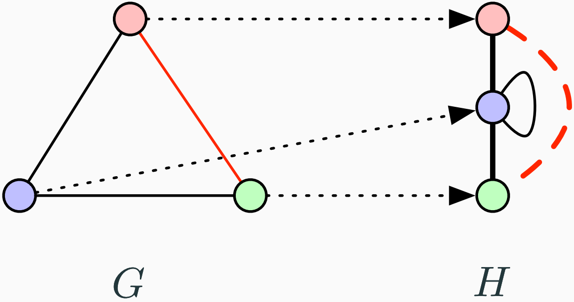
Does the graph have 3 components?



* additional surjectivity is required.

Measure how far away from homomorphism

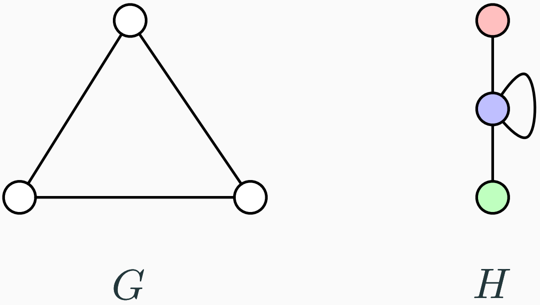
The edge not mapped to an edge is a violating edge.



The violation of a map is the number of violating edges.

Surjective Minimum Violation. SVio(H)
Input: $G = (V, E)$.

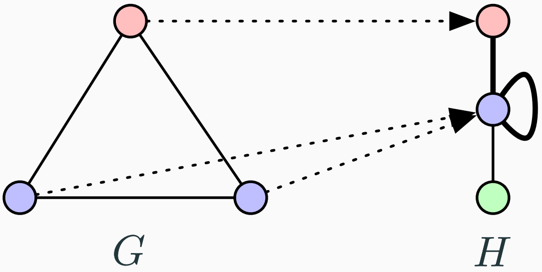
Output: A surjective map from G to H with minimum violation.



Surjective Minimum Violation. SVIO(H)

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

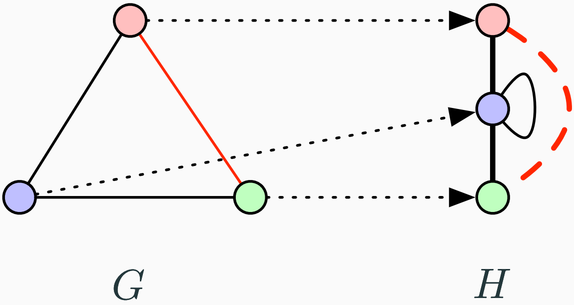
Output: A surjective map from G to H with minimum violation.



Surjective Minimum Violation. SVio(H)

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

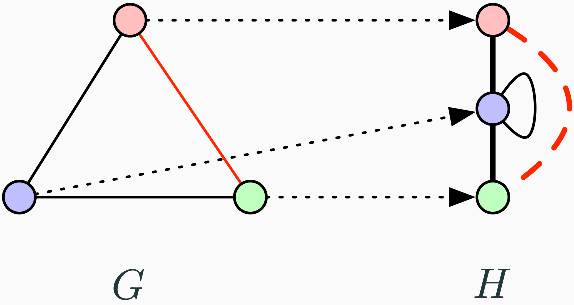
Output: A surjective map from G to H with minimum violation.



Surjective Minimum Violation. SVio(H)

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

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



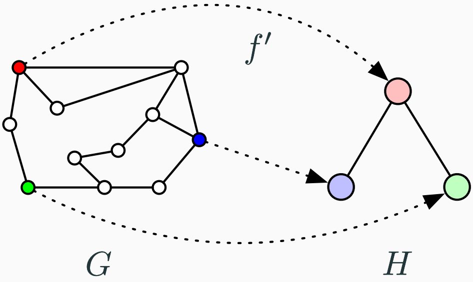
H is s-tractable if SVio(H) is tractable.

Retraction minimum violation. RVIO(H)

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

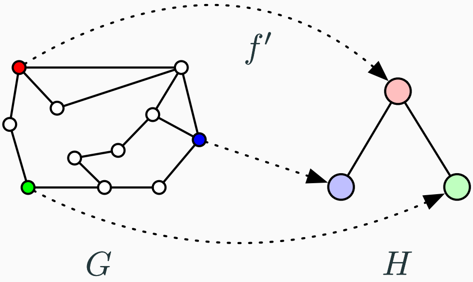
Output: A map f from G to H such that $f|_{V'} = f'$ and the violation is minimized.

Vertices in V' are called terminals.



Retraction minimum violation. 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 called terminals.



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

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

Why?

For a cut problem, there is usually a pattern graph H, such that

Fixed-Terminal Min Cut $\in$ **P**	$\Leftrightarrow$	H is r-tractable
Global Min Cut $\in$ **P**	$\Leftrightarrow$	H is s-tractable

Consequence: A unified tool to quickly decide if a cut problem is easy or hard by looking at the pattern graph H.

Out Results

- Relating r-tractability and s-tractability with various cut problems.
- A **complete classification** of r-tractable graphs.
- disconnected reflexive s-tractable graphs are defined by the s-tractability of its components.
- A **complete classification** of s-tractability of **trees**.

Out Results

- Relating r-tractability and s-tractability with various cut problems.
- A **complete classification** of r-tractable graphs.
- disconnected reflexive s-tractable graphs are defined by the s-tractability of its components.
- A **complete classification** of s-tractability of **trees**.

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.

Previous Work

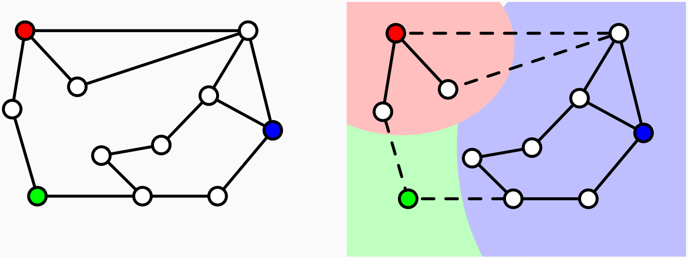
- Classification of s-tractable graphs and r-tractable graphs was studied under the name "G_c-cut". [Elem, Hassin & Monnot 13]
- A more general problem called 0-extension problem was studied, but only approximation was considered [Calinescu et. al. 01]

Modeling cut problems

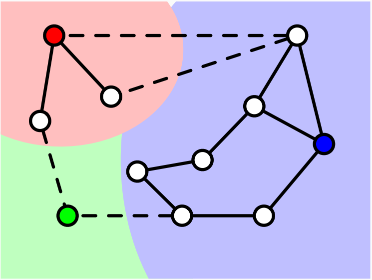
Problem: Min k-way cut

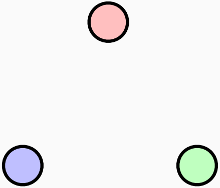
Input: G and the terminals $v_1, \ldots, v_k \in V(G)$

Output: Find a minimum set of edges F, such that such in $G - F$, each pair of terminals cannot reach each other.

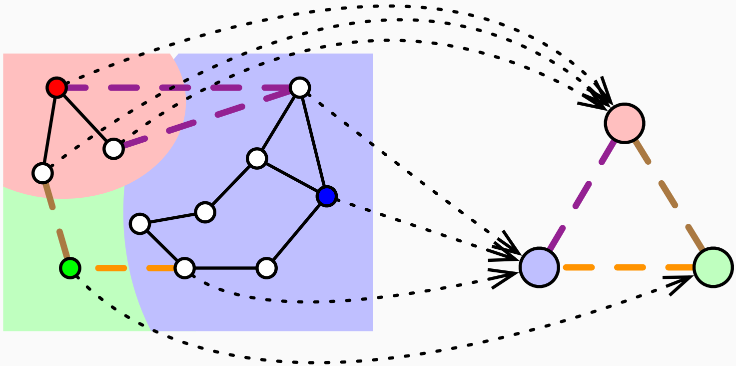


3-way cut





3-way cut



k-cut

Problem: min k-cut

Input: $G = (V, E)$

Output: A minimum set of edges F, such that $G - F$ has at least k components.

k-way cut and k-cut

I_k is the reflexive graph of k isolated vertices.

k-way cut is equivalent to $\textsc{RVio}(I_k)$.

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

k-cut is equivalent to $\textsc{SVio}(I_k)$.

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

Theorem
I_k is r-tractable if and only if $k \leq 2$. I_k is s-tractable for all k.

(ℓ, k)-way-cut

A set of edges F is a (ℓ, k)-way-cut for a set of terminals $v_1, \ldots, v_k$, if in $G - F$, the pairwise distance of the terminals is at least $\ell + 1$.

(ℓ, k)-way-cut

A set of edges F is a (ℓ, k)-way-cut for a set of terminals $v_1, \ldots, v_k$, if in $G - F$, the pairwise distance of the terminals is at least $\ell + 1$.

(∞, k)-way-cut is the standard k-way-cut.

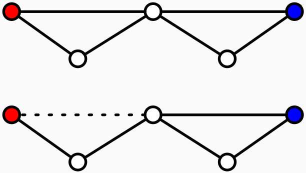
(ℓ, k)-way-cut

A set of edges F is a (ℓ, k)-way-cut for a set of terminals $v_1, \ldots, v_k$, if in $G - F$, the pairwise distance of the terminals is at least $\ell + 1$.

(∞, k)-way-cut is the standard k-way-cut.

Problem: (ℓ, k)-way-cut
Input: G, terminals $v_1, \ldots, v_k$

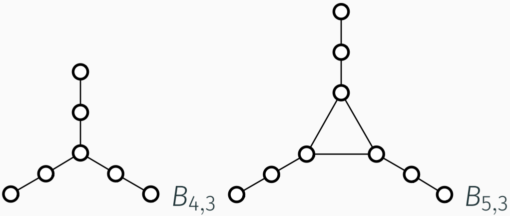
Output: A minimum cardinality (ℓ, k)-way-cut.



$(2, 2)$-cut

Theorem ([Mahjoub & McCormick 00])
$(\ell, 2)$-way-cut is tractable if and only if $\ell \leq 3$.

Theorem
(ℓ, k)-way-cut is equivalent to **RVIO**$(B_{\ell,k})$.



Cut problem in directed graphs

F is a set of edges. F is a

- k-reach-cut, if in $G - F$, there exists a set of k terminals, such that every vertex can reach at most one of the terminals.

Cut problem in directed graphs

F is a set of edges. F is a

- k-reach-cut, if in $G - F$, there exists a set of k terminals, such that every vertex can reach at most one of the terminals. In P.

Cut problem in directed graphs

F is a set of edges. F is a

- k-reach-cut, if in $G - F$, there exists a set of k terminals, such that every vertex can reach at most one of the terminals. In P.
- Linear-k-cut, if in $G - F$, there are terminals $v_1, \ldots, v_k$, such that v_i cannot reach v_j if $i < j$.

Cut problem in directed graphs

F is a set of edges. F is a

- k-reach-cut, if in $G - F$, there exists a set of k terminals, such that every vertex can reach at most one of the terminals. **In P**.
- Linear-k-cut, if in $G - F$, there are terminals $v_1, \ldots, v_k$, such that v_i cannot reach v_j if $i < j$. **Unknown if NP-hard**.

Cut problem in directed graphs

F is a set of edges. F is a

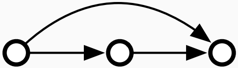
- k-reach-cut, if in $G - F$, there exists a set of k terminals, such that every vertex can reach at most one of the terminals. **In P**.
- Linear-k-cut, if in $G - F$, there are terminals $v_1, \ldots, v_k$, such that v_i cannot reach v_j if $i < j$. **Unknown if NP-hard**.
- Bicut, if in $G - F$, there are two vertices s and t that cannot reach each other.

Cut problem in directed graphs

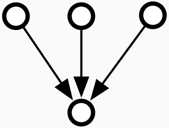
F is a set of edges. F is a

- k-reach-cut, if in $G - F$, there exists a set of k terminals, such that every vertex can reach at most one of the terminals. **In P**.
- Linear-k-cut, if in $G - F$, there are terminals $v_1, \ldots, v_k$, such that v_i cannot reach v_j if $i < j$. **Unknown if NP-hard**.
- Bicut, if in $G - F$, there are two vertices s and t that cannot reach each other. **Unknown if NP-hard**.

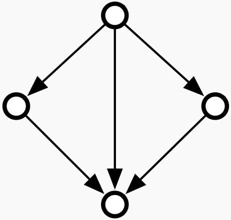
Cut problem in directed graphs



(a) T_3, lineat-3-cut



(b) S_k, $k = 3$, 3-reach-cut



(c) H_{bicut}, bicut

Previous cut problem is equivalent to **SVIO**(H) for some directed graph H.

Classification of r-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: CVIO(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: CVIO(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.

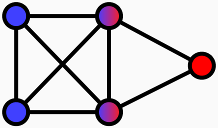
H is c-tractable if CVIO(H) is tractable.

Minimum cost and violation

Problem: CVio(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 CVio(H) is tractable.

Theorem ([Deineko et.al. 08])
H is c-tractable if and only if there are two sets A and B such that $A \cup B = V$, and $H[A]$ and $H[B]$ are cliques.



But why do we care about CVIO(H)?

RVIO(H) reduces to CVIO(H).

But why do we care about CVIO(H)?

RVIO(H) reduces to CVIO(H).

Fixing vertices using cost.

But why do we care about CVɪᴏ(H)?

RVɪᴏ(H) reduces to CVɪᴏ(H).

Fixing vertices using cost.

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

Input to CVɪᴏ(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 CVIO(H)?

RVIO(H) reduces to CVIO(H).

Fixing vertices using cost.

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

Input to CVIO(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 r-tractable are the same?

But why do we care about CVio(H)?

RVio(H) reduces to CVio(H).

Fixing vertices using cost.

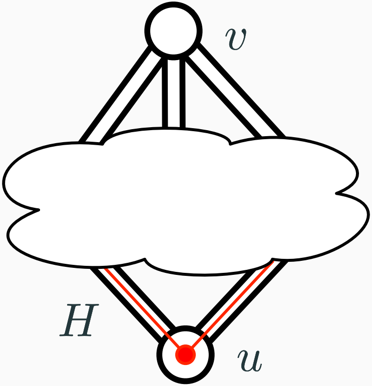
Input of RVio(H) is G and $f' : V' \to U$.

Input to CVio(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 r-tractable are the same?

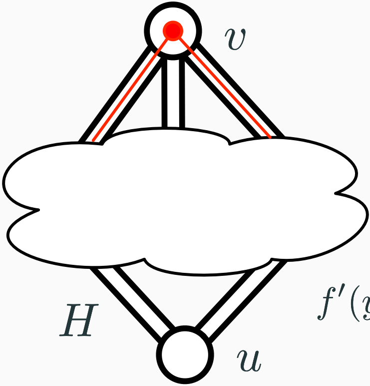
Nope: P_4 is r-tractable but c-tractable.

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.

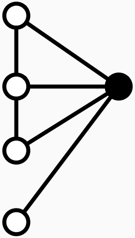
Superseded vertices

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 $u \prec v$.

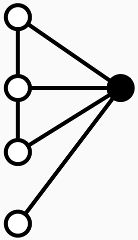
Apex

A vertex is an apex if no vertex supersedes it.



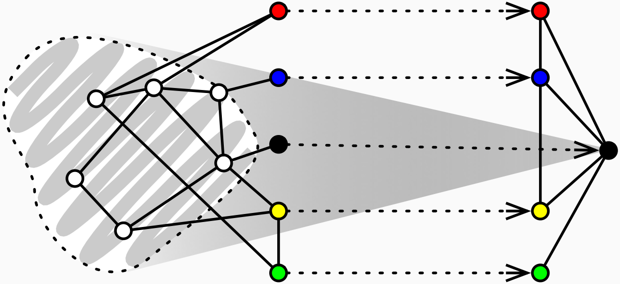
Apex

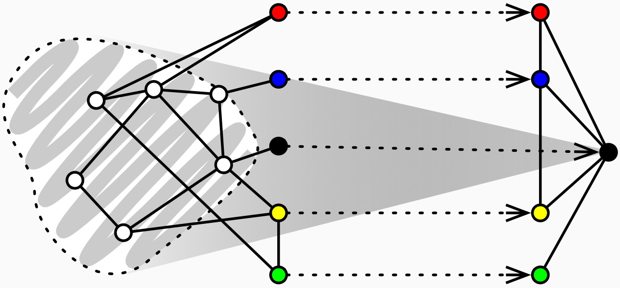
A vertex is an apex if no vertex supersedes it.



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 r-tractable.

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

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

Complete graphs are c-tractable.

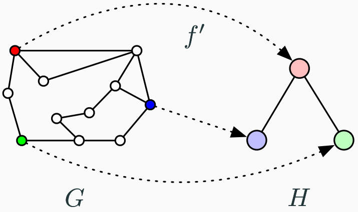
r-tractability and c-tractability

Theorem ([Kawarabayashi & X 20])
H is r-tractable if the apex subgraph of H is c-tractable.

$\mathbf{RVio}(H)$

$G = (V, E)$

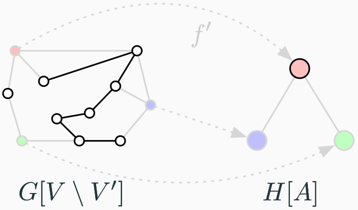
$f' : V' \to U$



$\mathbf{CVio}(H[A])$

$G[V \setminus V']$

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



Classification of RVIO

The harder direction requires knowledge from CSP theory.

Theorem ([Kawarabayashi & X 20])
H is r-tractable ***if and only if*** *the apex subgraph of H is c-tractable.*

Classification of RVIO

The harder direction requires knowledge from CSP theory.

Theorem ([Kawarabayashi & X 20])
H is r-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.

Consequence

A strange problem

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

Output: A minimum cardinality set of edges F such that

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

Consequence

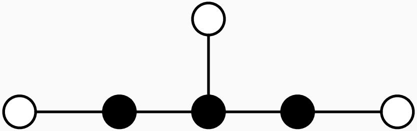
A strange problem

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 **RVIO**(H), where H is:



Consequence

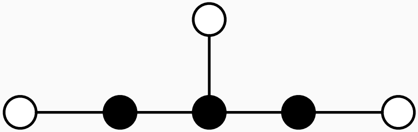
A strange problem

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 **RVio**(H), where H is:



- $(3, 3)$-way-cut is NP-hard.
- $(4, 2)$-way-cut is NP-hard.

s-tractable graphs

Previously known

SHom(H)
Input: Graph G.

Output: Decide if there is a surjective homomorphism from G to H.

A graph H is s_0-tractable if **SHom**(H) is tractable.

- H is r-tractable then it is s-tractable.
- H is not s_0-tractable then it is not s-tractable.

[Elem, Hassin & Monnot 13]

Tractability from components

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

Tractability from components

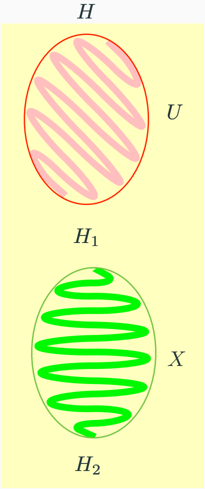
Theorem
A reflexive graph H is s-tractable if and only if each of its component is s-tractable.

We will prove the case when H has two components.

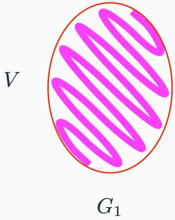
H is composed of components H_1 and H_2.

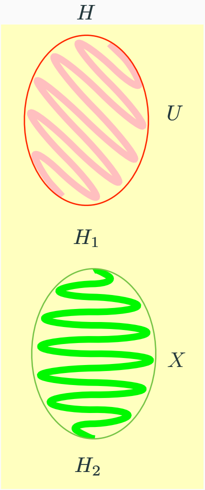
H_1 is not s-tractable. We reduce SVIO(H_1) to SVIO(H).

Hardness

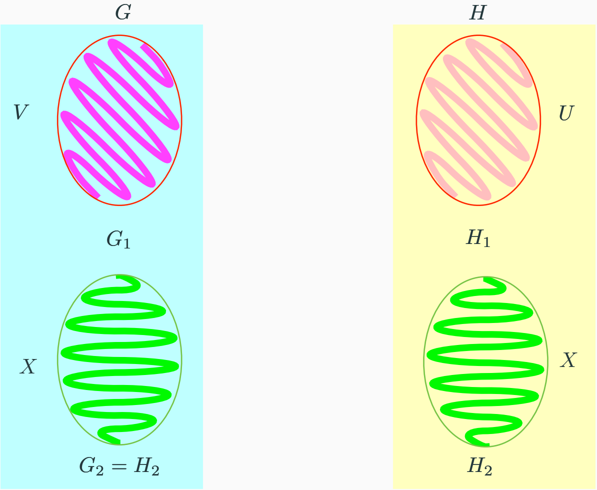


Hardness

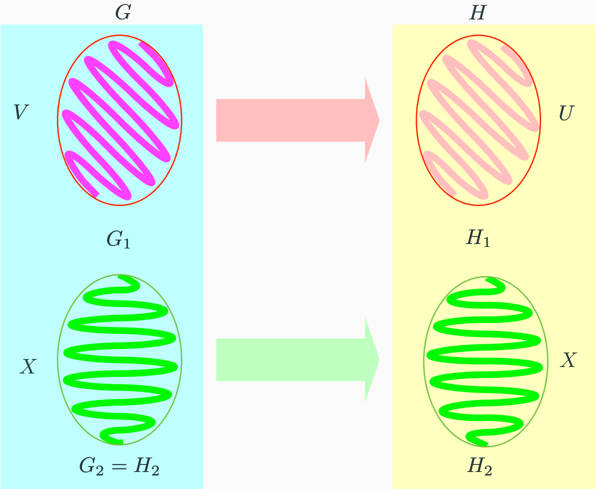




Hardness



Hardness



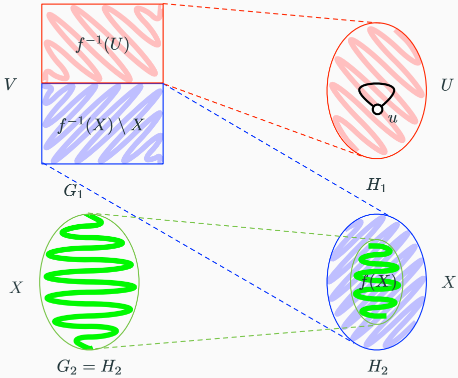
Hardness

Proof idea: 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,
- $f'(X) = X$,
- $f'(V) = U$.

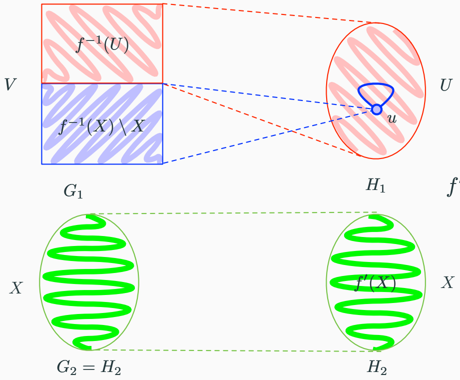
$f'|_V$ is the desired minimum violation map from G_1 to H_1.

Hardness



$$f(X) \subset X$$

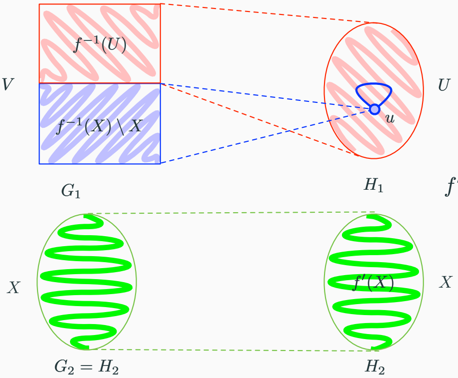
Hardness



$$f(X) \subset X$$

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

Hardness

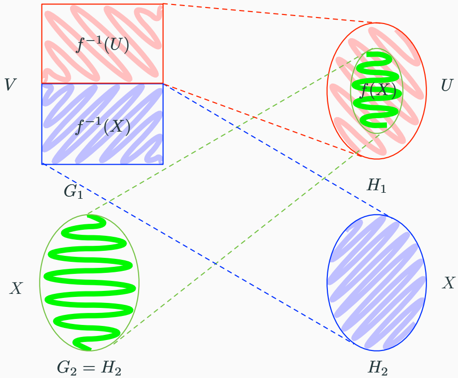


$$f(X) \subset X$$

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

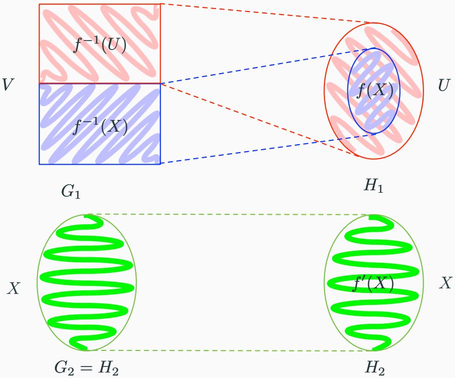
Reflexivity is crucial.

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.

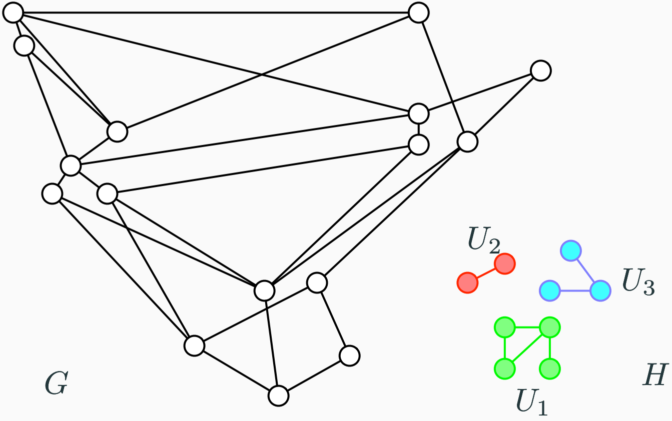
Set up

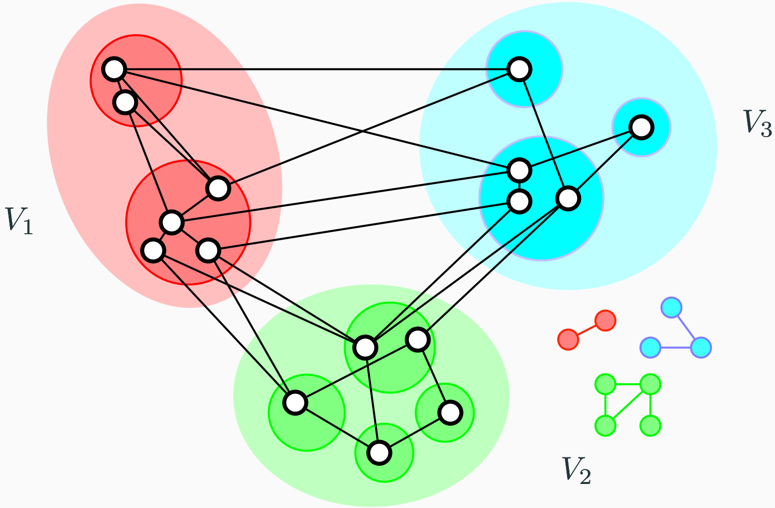
H is a k vertex graph consist of components $U_1, \ldots, U_\ell$.

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

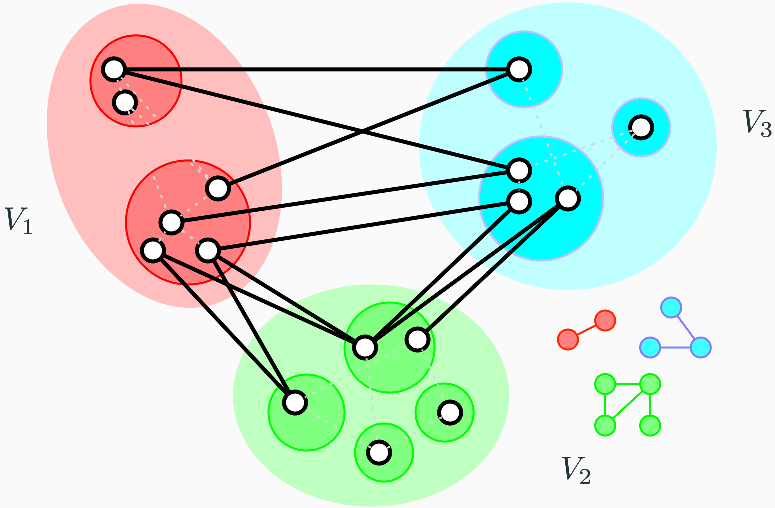
f is the optimal solution of SVIO(H) with input graph G.

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


U_2
U_3
U_1
G
H



V_1
V_2
V_3



V_1
V_3
V_2

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

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

Enumerating ℓ-cuts with small value

Theorem
There exists an algorithm that takes $n^{O(k)}$ time and produce all ℓ-cuts with value at most the value of a min-k-cut.

Enumerating ℓ-cuts with small value

Theorem
There exists an algorithm that takes $n^{O(k)}$ time and produce all ℓ-cuts with value at most the value of a min-k-cut.

Through spanning tree packing [Thorup 08; Chekuri, Quanrud, X 20].

Algorithm

Input graph G.

1. For each ℓ-partition $(V_1, \dots, V_\ell)$ of value at most min-k-cut
    1.1 Solve SVIO($H[U_i]$) with input $G[V_i]$.
    1.2 Combine the solutions into a candidate solution.
2. Output the minimum candidate solution.

Main Theorem

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

Theorem
A reflexive tree T is s-tractable ***if and only if*** $\operatorname{diam}(T) \leq 4$ *and for every* 3 *distinct vertices at least 2 has distance at most* 3*.*

Trees

Theorem
A reflexive tree T is s-tractable ***if and only if*** $\text{diam}(T) \leq 4$ *and for every* 3 *distinct vertices at least 2 has distance at most* 3.

Proof idea: Hardess

- For trees the distance relations completely determines the existance of homomorphism.
- We design gadgets to force distance relations of fixed vertices.
- Conclude hardness from $(3, 3)$-way-cut and $(4, 2)$-way-cut.

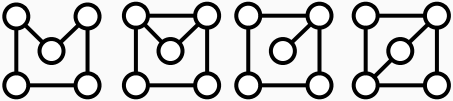
For the trees not covered by hardness, we see that T is r-tractable, therefore T is s-tractable.

Open Problems

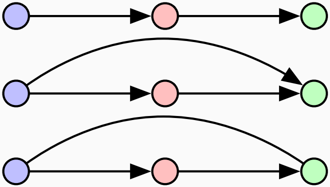
Classify the s-tractable graphs

All 4 vertex reflexive graphs and 2 vertex digraphs have been characterized.

- Conjectures
  - H has a surjective homomorphism to H', and H' is not s-tractable, then H is not s-tractable.
- 5 vertex graphs?



- 3 vertex digraphs?



Thank you