Flow-augmentation:

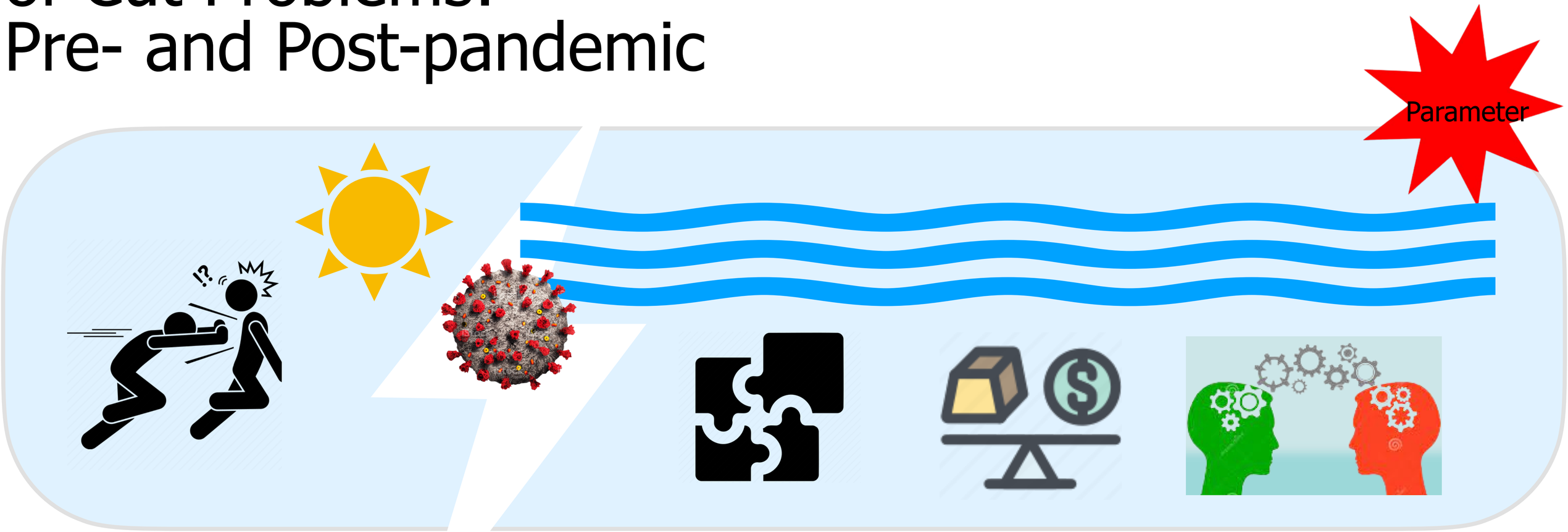
advances in parameterized (weighted) cut problems

IIT Palakkad
August 07, 2023

Roohani Sharma

max planck institut
informatik

Parameterized Complexity Landscape of Cut Problems: Pre- and Post-pandemic



Greedy

Non-Greedy

Cut problems of our interest

The greedy era (pre-pandemic)

- Tools:
  - Important Cuts
  - Shadow Removal
- PC Cut landscape
- Open Questions

Flow-augmentation (the pandemic)

- What is it?
- Easy Application:
  - Weighted s-t cut
- Other applications:
  - ℓ-Chain SAT

MinCSPs ("Post"-pandemic)

Boolean MinCSP dichotomy

Weighted Cut Problems ("Post"-pandemic)

- Weighted Edge Multiway Cut
- Weighted Edge Multicut
- Weighted Vertex Multicut
- Weighted (Subset) DFAS

Cut problems of our interest

The greedy era (pre-pandemic)

- Tools:
  - Important Cuts
  - Shadow Removal
- PC Cut landscape
- Open Questions

Flow-augmentation (the pandemic)

- What is it?
- Easy Application:
  - Weighted s-t cut
- Other applications:
  - ℓ-Chain SAT

MinCSPs ("Post"-pandemic)

Boolean MinCSP dichotomy

Weighted Cut Problems ("Post"-pandemic)

- Weighted Edge Multiway Cut
- Weighted Edge Multicut
- Weighted Vertex Multicut
- Weighted (Subset) DFAS

Cut Problems

s-t Cut

Input:
(Directed) graph G,
a pair of vertices (s, t) (terminals)
positive integer k
(or a weight function weight wt : $E(G) \to \mathbb{Z}^+$ and positive weight budget W)

Question:
?$\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$ (or $\text{wt}(Z) \leq W$), and
$G - Z$ has no $s \to t$ path.

Polynomial-time solvable

Cut Problems

(Directed) Multicut

(Directed) graph G,
pairs of vertices (terminals) $(s_1, t_1), \ldots, (s_p, t_p)$
positive integer k

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, and
for each $i \in \{1,\ldots,p\}$,
$G - Z$ has no $s_i \to t_i$ path.

Directed case: NP-hard for p=2
Undirected case: NP-hard for p=3

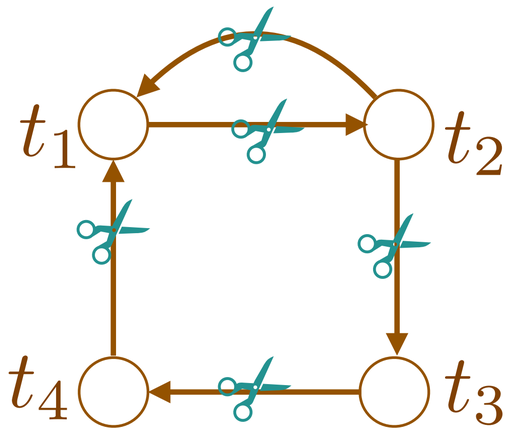
Cut Problems

(Directed) Multicut

(Directed) graph G,
pairs of vertices (terminals) $(s_1, t_1), \ldots, (s_p, t_p)$
positive integer k

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, and
for each $i \in \{1,\ldots,p\}$,
$G - Z$ has no $s_i \to t_i$ path.

Directed case: NP-hard for p=2
Undirected case: NP-hard for p=3



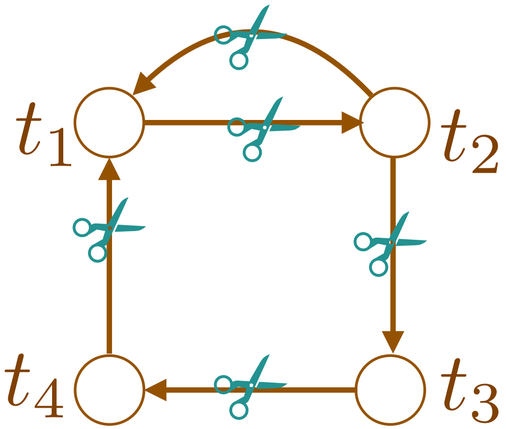
Cut Problems

(Directed) Multicut

(Directed) graph G,
pairs of vertices (terminals) $(s_1, t_1), \ldots, (s_p, t_p)$
positive integer k

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, and
for each $i \in \{1,\ldots,p\}$,
$G - Z$ has no $s_i \to t_i$ path.

Directed case: NP-hard for p=2
Undirected case: NP-hard for p=3

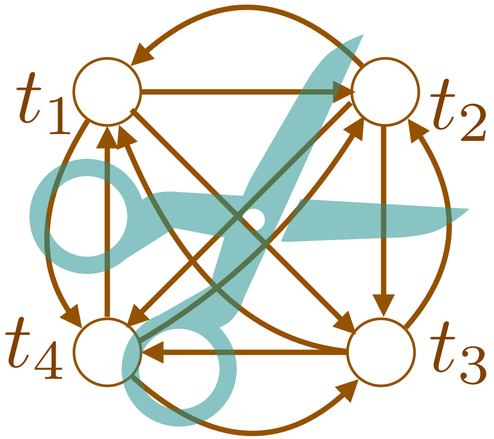


(Directed) MultiwayCut

(Directed) graph G,
vertices (terminals) $t_1, \ldots, t_p$
positive integer k

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, and
$G - Z$ has no $t_i \to t_j$ path
for any $i, j \in \{1,\ldots,p\}$, $i \neq j$.

Directed case: NP-hard for p=2
Undirected case: NP-hard for p=3



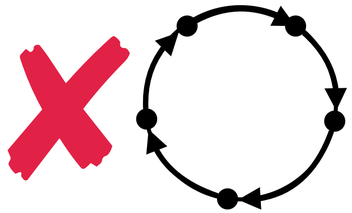
Cut Problems

Directed Feedback Arc Set (DFAS)

Directed graph G,
positive integer k

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, and
$G - Z$ has no directed cycle.

NP-hard

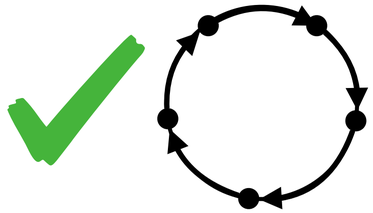


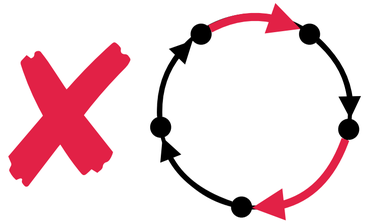
Subset DFAS

Directed graph G,
Red arcs $R \subseteq E(G)$,
positive integer k

$?\exists$ a set $Z \subseteq E(G) \setminus R$
such that $|Z| \leq k$, and
$G - Z$ has no directed cycle with at least one red arc.

NP-hard





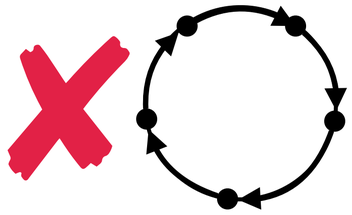
Cut Problems

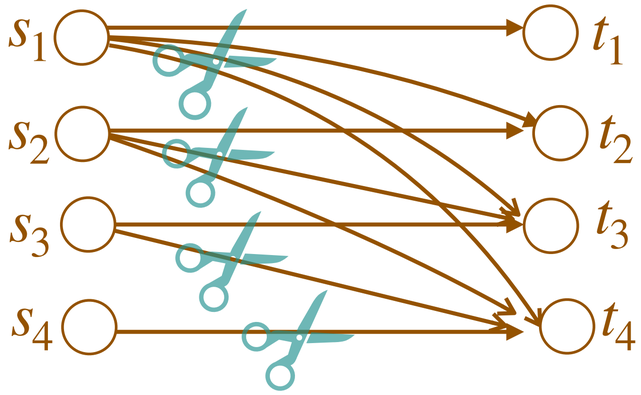
Directed Feedback Arc Set (DFAS)

Directed graph G,
positive integer k

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, and
$G - Z$ has no directed cycle.

NP-hard



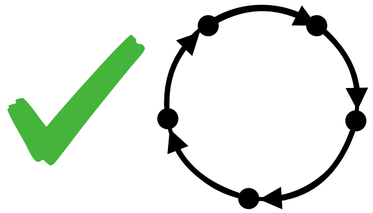


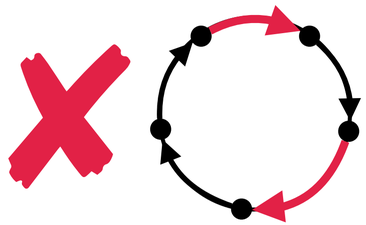
Subset DFAS

Directed graph G,
Red arcs $R \subseteq E(G)$,
positive integer k

$?\exists$ a set $Z \subseteq E(G) \setminus R$
such that $|Z| \leq k$, and
$G - Z$ has no directed cycle with at least one red arc.

NP-hard

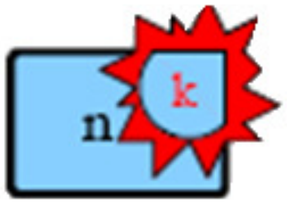




Parameterized Complexity

— Multivariate analysis of NP-hard problems

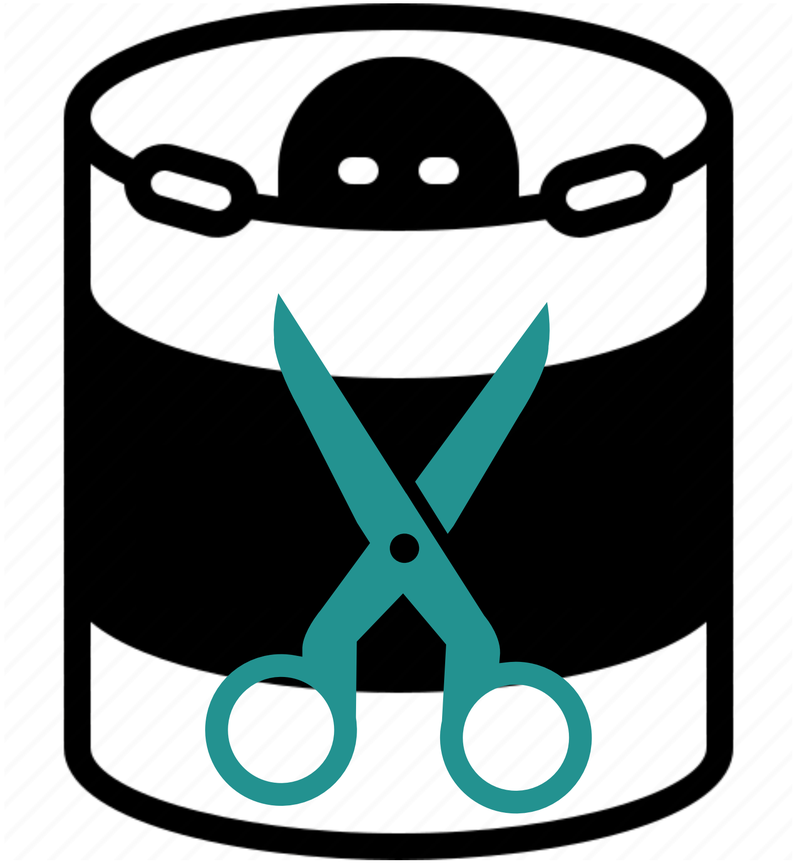
— Parameter is solution size k



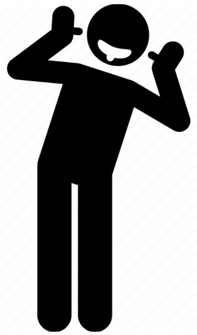
FPT: $f(k) \cdot n^{\mathcal{O}(1)}$

W[1]-hard, otherwise.

Cut problems remained elusive for a very long time!



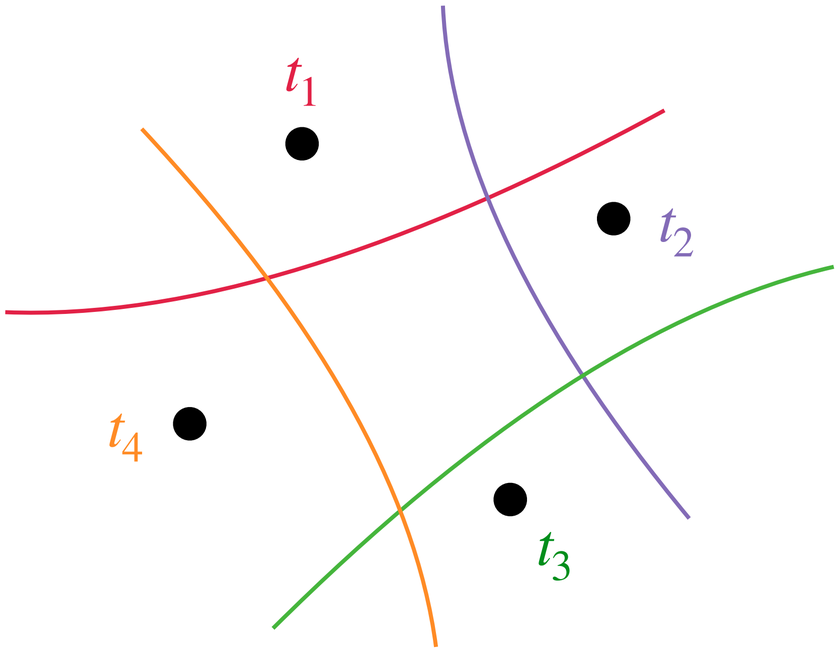
Even the simplest algorithm is based on a novel concept of **important cuts**.



Naive algorithm

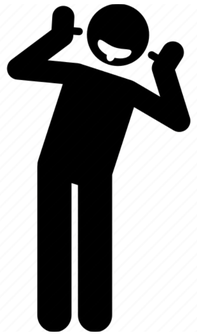
UNDIRECTED MULTIWAYCUT

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.



Any solution contains some:

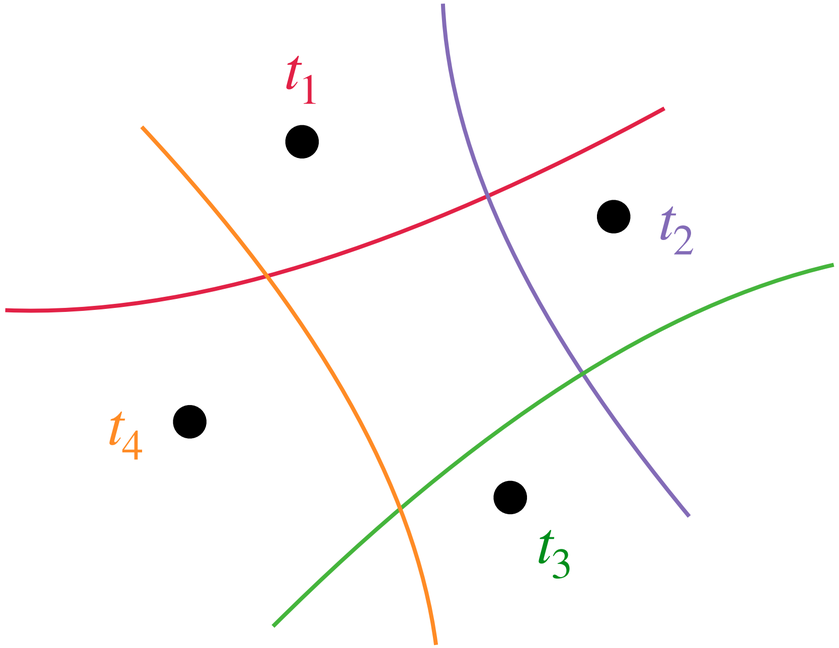
- $t_1 - T \backslash t_1$ minimal cut
- $t_2 - T \backslash t_2$ minimal cut
- $t_3 - T \backslash t_3$ minimal cut
- $t_4 - T \backslash t_4$ minimal cut



Naive algorithm

UNDIRECTED MULTIWAYCUT

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.



Any solution contains some:

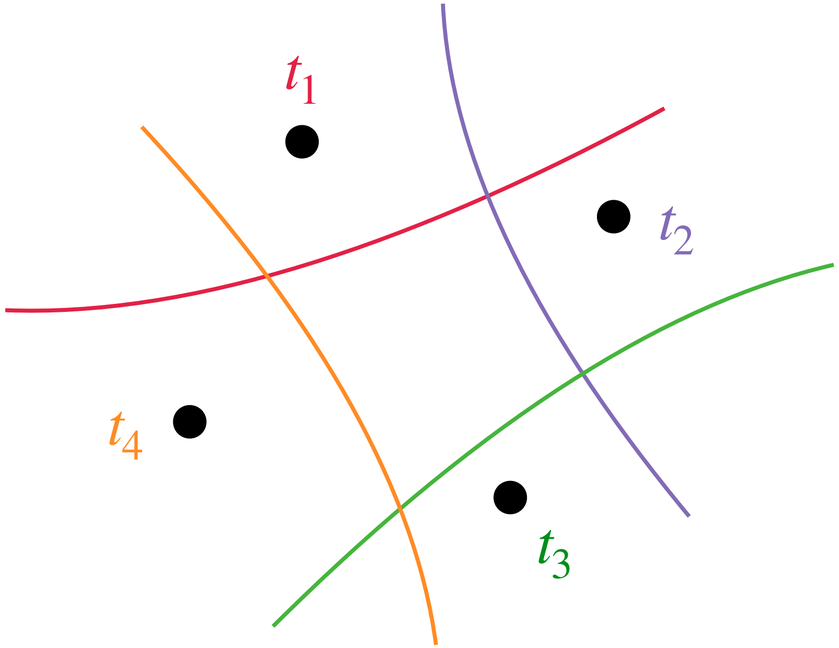
- $t_1 - T \backslash t_1$ minimal cut
- $t_2 - T \backslash t_2$ minimal cut
- $t_3 - T \backslash t_3$ minimal cut
- $t_4 - T \backslash t_4$ minimal cut

Z is a minimal (X, Y)-cut,
if no proper subset of Z is an (X, Y)-cut.

Naive algorithm

UNDIRECTED MULTIWAYCUT

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.



Any solution contains some:

- $t_1 - T \backslash t_1$ minimal cut
- $t_2 - T \backslash t_2$ minimal cut
- $t_3 - T \backslash t_3$ minimal cut
- $t_4 - T \backslash t_4$ minimal cut

Z is a minimal (X, Y)-cut,
if no proper subset of Z is an (X, Y)-cut.

Branch

Naive algorithm

UNDIRECTED MULTIWAYCUT

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.

Naive algorithm

1. If each $t_i \in T$ is in a different connected component of G, then we are done.
2. If there exists $t_i \in T$, such that there is a path from t_i to $T \backslash t_i$, then **enumerate every minimal** $(t_i, T \backslash t_i)$**-cut** of size at most k, and branching of choosing one such cut S in the solution.
3. Set $G := G \backslash S$ and $k := k - |S|$.
4. Go to Step 1.

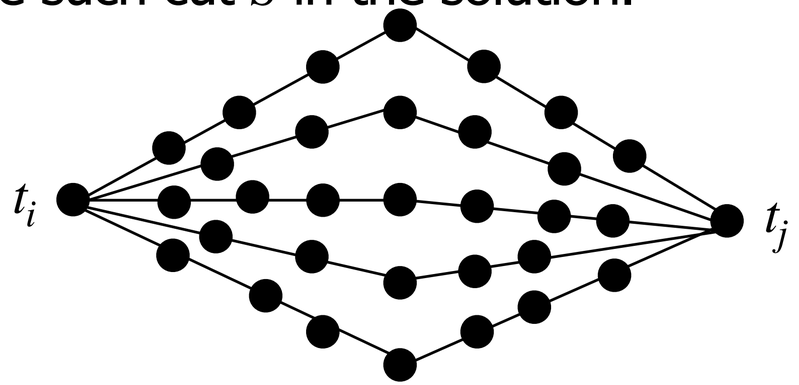
Naive algorithm

UNDIRECTED MULTIWAYCUT

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.

Naive algorithm

1. If each $t_i \in T$ is in a different connected component of G, then we are done.
2. If there exists $t_i \in T$, such that there is a path from t_i to $T \backslash t_i$, then **enumerate every minimal** $(t_i, T \backslash t_i)$**-cut** of size at most k, and branching of choosing one such cut S in the solution.
3. Set $G := G \backslash S$ and $k := k - |S|$.
4. Go to Step 1.



Naive algorithm

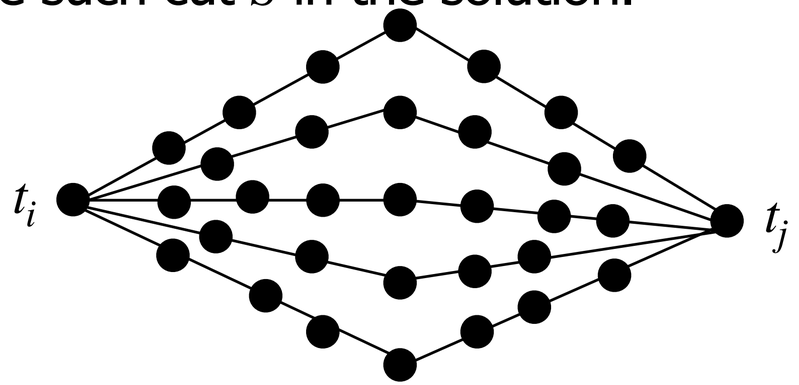
UNDIRECTED MULTIWAYCUT

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.

Naive algorithm

1. If each $t_i \in T$ is in a different connected component of G, then we are done.
2. If there exists $t_i \in T$, such that there is a path from t_i to $T \backslash t_i$, then **enumerate every minimal $(t_i, T \backslash t_i)$-cut** of size at most k, and branching of choosing one such cut S in the solution.
3. Set $G := G \backslash S$ and $k := k - |S|$.
4. Go to Step 1.

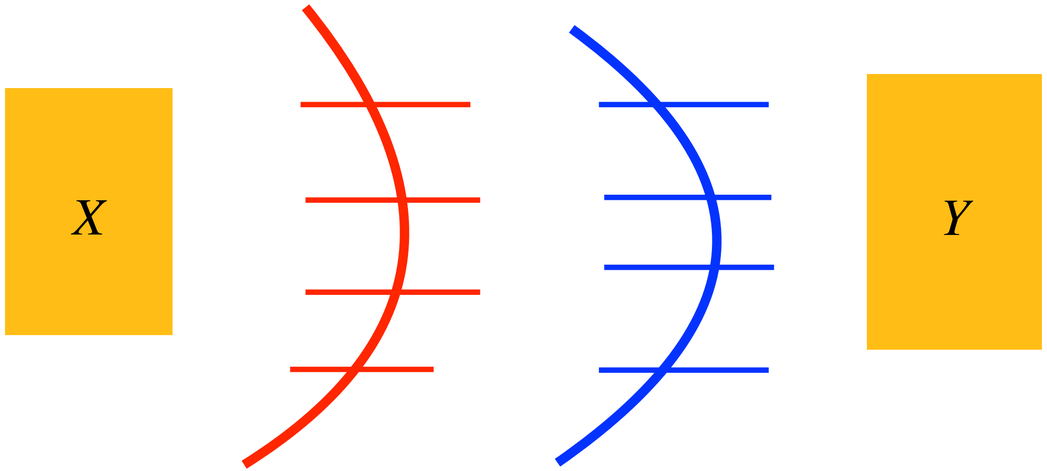
$$n^{\mathcal{O}(k)}$$



Important Cuts: Directed and Undirected

Daniel Marx

Daniel Marx (2004) defined a partial order that compares two minimal X-Y cuts.



Blue is better than red.

Important Cuts

Daniel Marx

For any $R \subseteq V(G)$, let $\delta(\mathbf{R})$ is the set of edges with exactly one endpoint in R.

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

Important Cuts

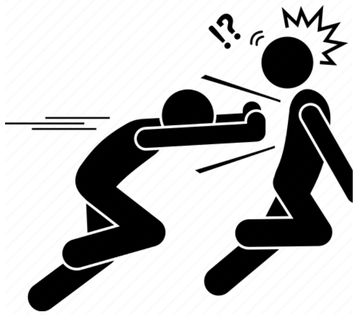
Daniel Marx

For any $R \subseteq V(G)$, let $\delta(\mathbf{R})$ is the set of edges with exactly one endpoint in R.

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

A minimal $X - Y$ cut $\delta(R')$ is **better** than the $X - Y$ cut $\delta(R)$ if:

- $R \subset R'$, and
- $|\delta(R')| \leq |\delta(R)|$.



Important Cuts

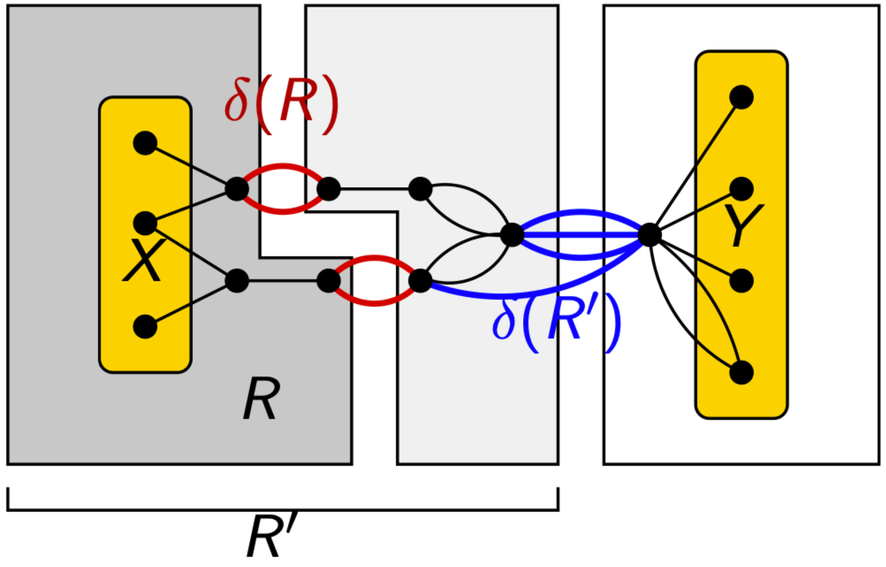
Daniel Marx

For any $R \subseteq V(G)$, let $\delta(\mathbf{R})$ is the set of edges with exactly one endpoint in R.

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

A minimal $X - Y$ cut $\delta(R')$ is **better** than the $X - Y$ cut $\delta(R)$ if:

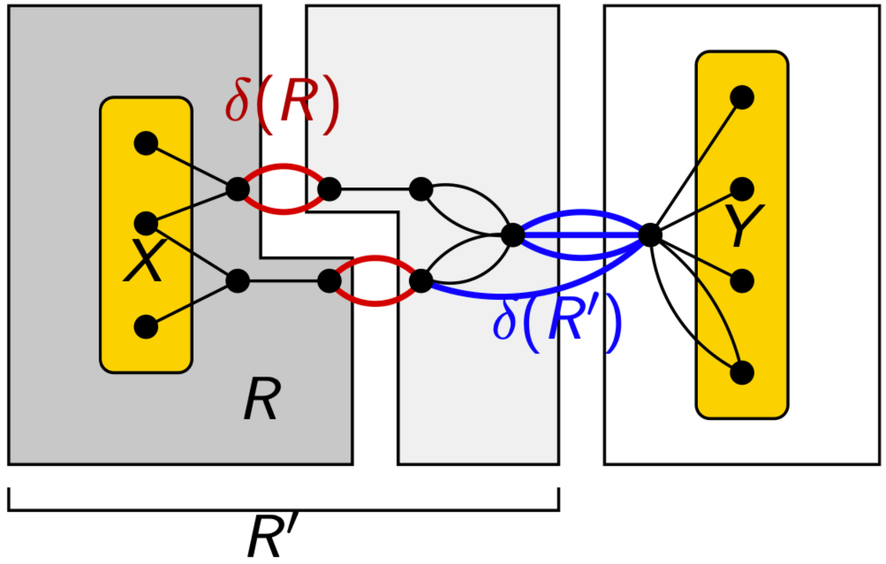
- $R \subset R'$, and
- $|\delta(R')| \leq |\delta(R)|$.



Important Cuts

Daniel Marx

Daniel Marx (2004) defined a partial order that compares two minimal $X - Y$ cuts.

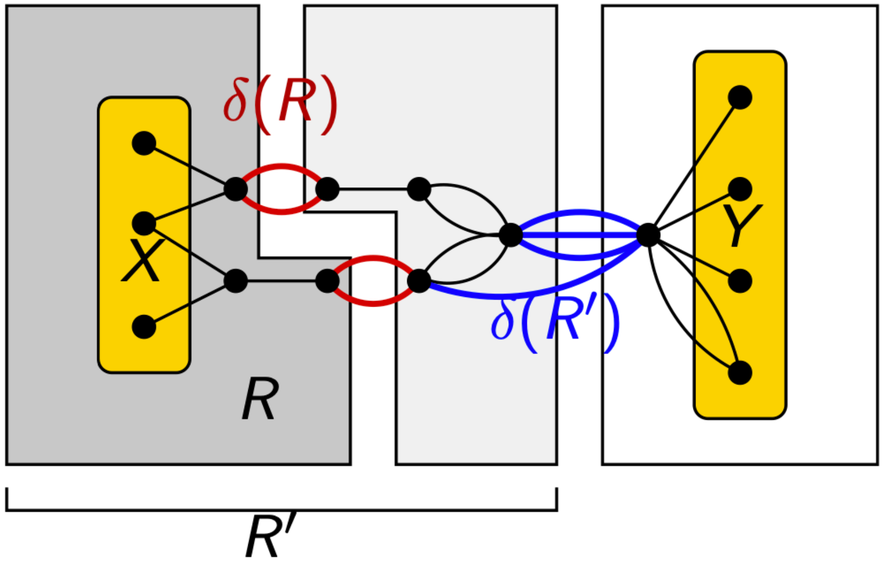


He called the maximum elements of this partial order as **important $\mathbf{X - Y}$ cuts.**

Important Cuts

Daniel Marx

Daniel Marx (2004) defined a partial order that compares two minimal $X - Y$ cuts.

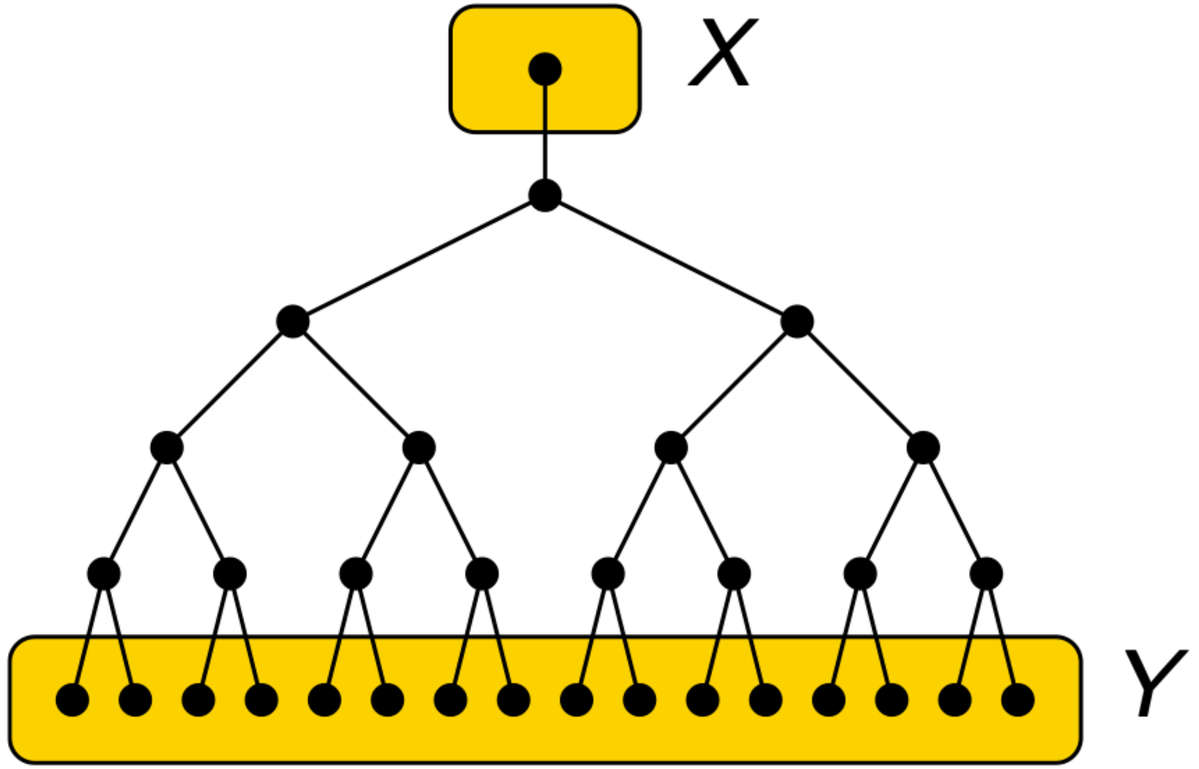


He called the maximum elements of this partial order as **important $\mathbf{X - Y}$ cuts.**

Number of **important $\mathbf{X - Y}$ cuts** of size at most k is at most 4^k.
Can be enumerated in $\mathcal{O}(4^k \cdot k \cdot (n+m))$

The 4^k bound is tight!

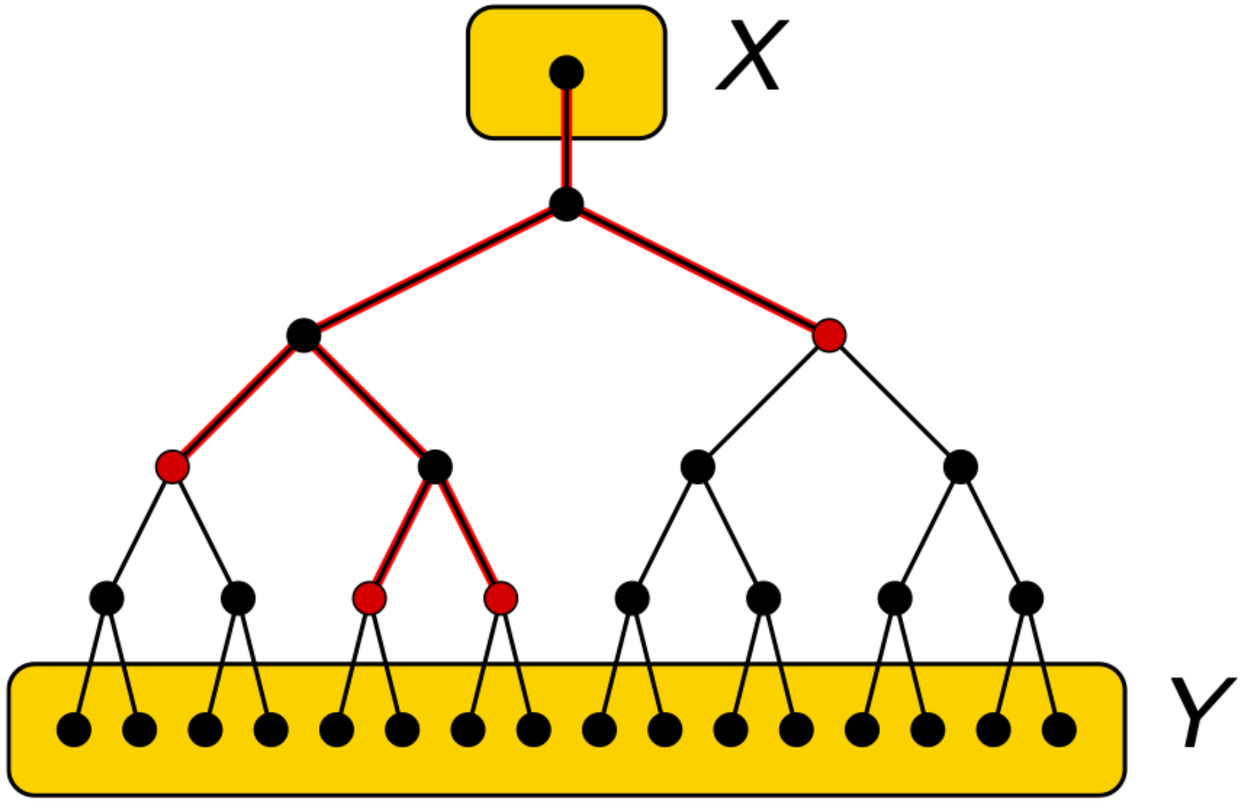
Daniel Marx



A rooted complete binary tree with $n > k$ levels.

The 4^k bound is tight!

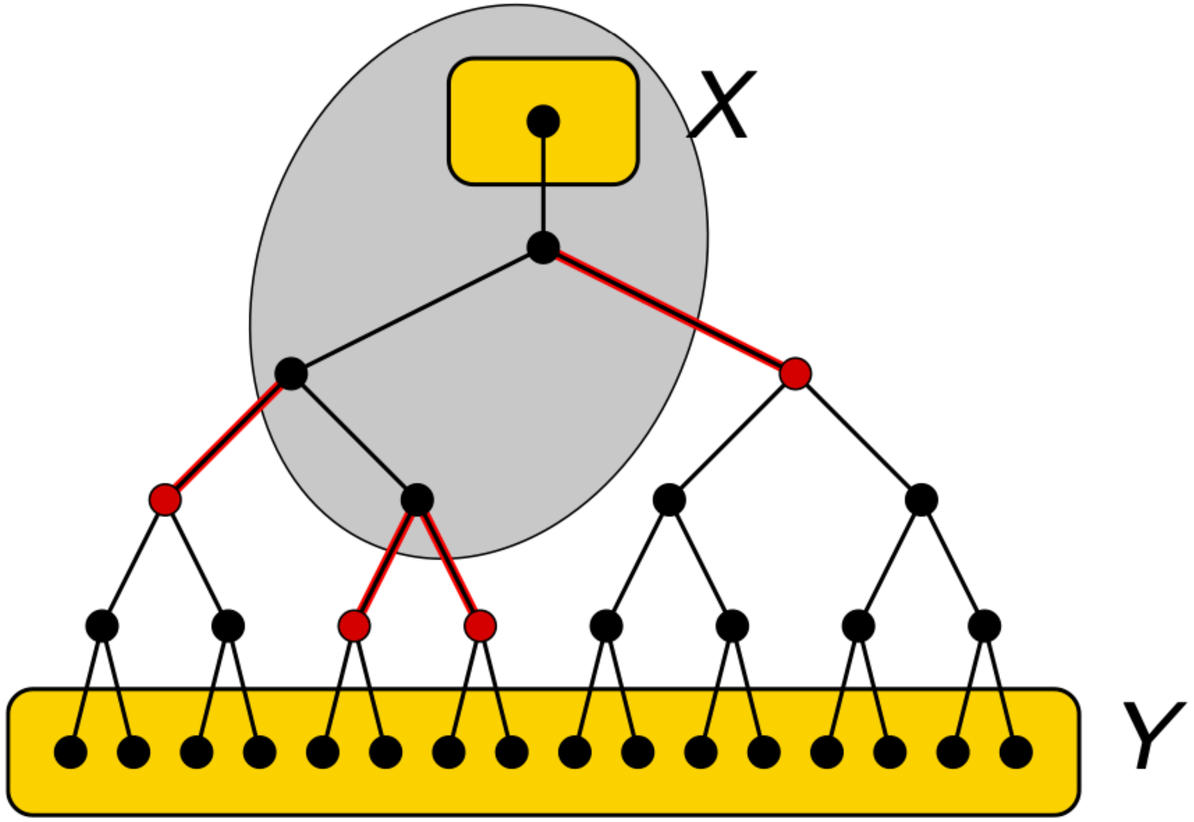
Daniel Marx



Any subtree with k leaves, implies an important $X - Y$ cut of size at most k.

The 4^k bound is tight!

Daniel Marx



Number of subtrees with k leaves is the Catalan number.

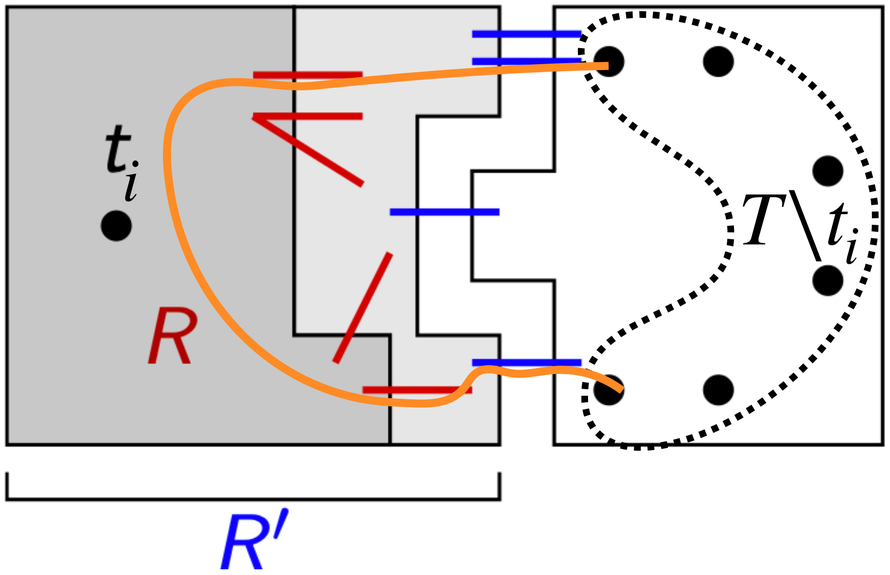
$$C_{k-1} = \frac{1}{k}\binom{2k-2}{k-1} \geq 4^k/\text{poly}(k)$$

Important Cuts:MWC

UNDIRECTED MULTIWAYCUT (MWC)

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.

Branch on **important** $t_i - T \backslash t_i$ separators of size at most k.



Part of solution Better than red

Important Cuts

Daniel Marx

UNDIRECTED MULTIWAYCUT (MWC)

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.

1. If each $t_i \in T$ is in a different connected component of G, then we are done.
2. If there exists $t_i \in T$, such that there is a path from t_i to $T \backslash t_i$, then **enumerate every minimal** $(t_i, T \backslash t_i)$**-cut** of size almost k, and branching of choosing one such cut S in the solution.
3. Set $G := G \backslash S$ and $k := k - |S|$.
4. Go to Step 1.

important

Important Cuts

Daniel Marx

UNDIRECTED MULTIWAYCUT (MWC)

Undirected graph G
terminals $T = \{t_1, \ldots, t_p\}$
Delete k edges to kill all $t_i - t_j$ paths.

1. If each $t_i \in T$ is in a different connected component of G, then we are done.
2. If there exists $t_i \in T$, such that there is a path from t_i to $T \backslash t_i$, then **enumerate every minimal** $(t_i, T \backslash t_i)$**-cut** of size almost k, and branching of choosing one such cut S in the solution.
3. Set $G := G \backslash S$ and $k := k - |S|$.
4. Go to Step 1.

important

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

Important Cuts

Number of **important $X - Y$ cuts** of size at most k is at most 4^k.
Can be enumerated in $\mathcal{O}(4^k \cdot k \cdot (n+m))$

Inherently greedy: Replace a part of the solution with something that "cuts" as much as the original part.

Important Cuts

Number of **important $X - Y$ cuts** of size at most k is at most 4^k.
Can be enumerated in $\mathcal{O}(4^k \cdot k \cdot (n+m))$

Inherently greedy: Replace a part of the solution with something that "cuts" as much as the original part.

Undirected Multiway is FPT.

Directed Feedback Arc Set is FPT.

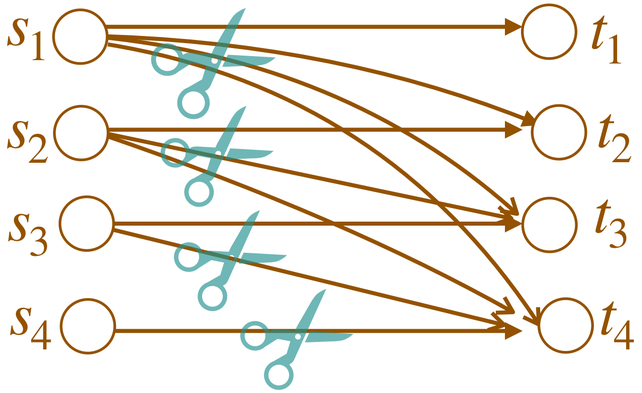
Important Cuts

Number of **important $X - Y$ cuts** of size at most k is at most 4^k.
Can be enumerated in $\mathcal{O}(4^k \cdot k \cdot (n+m))$

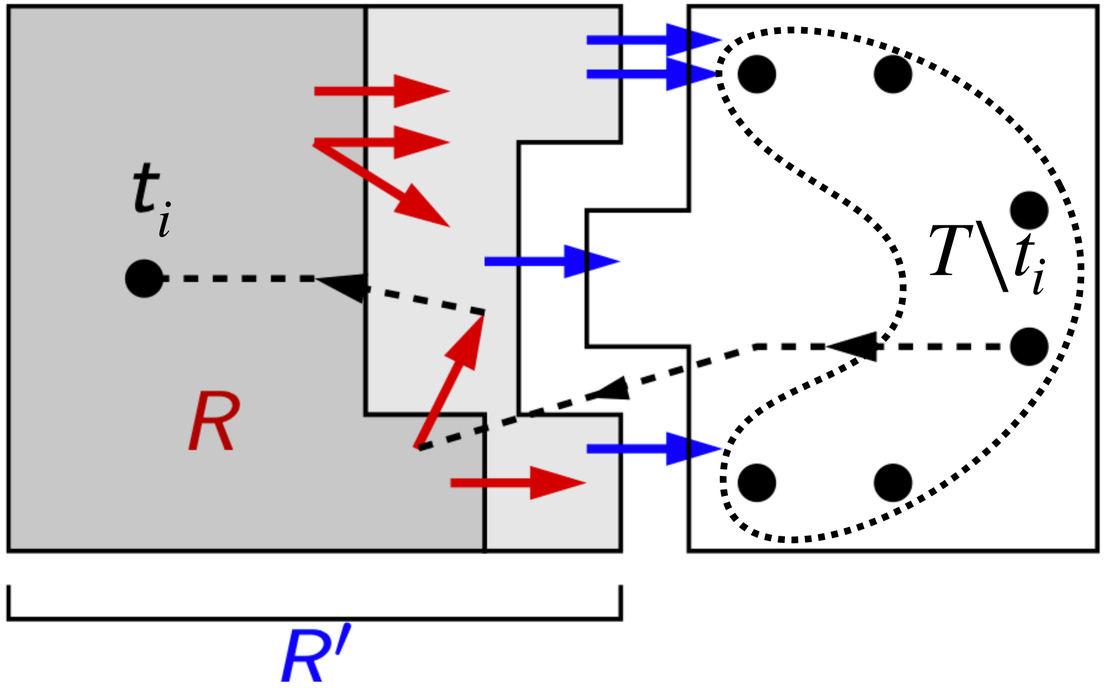
Inherently greedy: Replace a part of the solution with something that "cuts" as much as the original part.

Undirected Multiway is FPT.

Directed Feedback Arc Set is FPT.



Important Cuts based naive branching does **not** work for DIRECTED MULTIWAY CUT



Shadow Removal [STOC 2011]

Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

Daniel Marx

Igor Razgon

Shadow Removal [STOC 2011]

Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$

Shadowless solution w.r.t T:
$Z \subseteq V(G)$ is shadowless if
$\forall v \in V(G) \setminus Z$,
$\exists$ paths $T \rightsquigarrow v \rightsquigarrow T$

Daniel Marx

Igor Razgon

Shadow Removal [STOC 2011]

Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$

Shadowless solution w.r.t T:
$Z \subseteq V(G)$ is shadowless if
$\forall v \in V(G) \setminus Z$,
$\exists$ paths $T \rightsquigarrow v \rightsquigarrow T$

$t_i \rightsquigarrow v \rightsquigarrow t_j, i \neq j$

Daniel Marx

Igor Razgon

Shadow Removal [STOC 2011]

Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$

Shadowless solution w.r.t T:
$Z \subseteq V(G)$ is shadowless if
$\forall v \in V(G) \setminus Z$,
$\exists$ paths $T \rightsquigarrow v \rightsquigarrow T$

$t_i \rightsquigarrow v \rightsquigarrow t_j, i \neq j$ ✗

$t_i \rightsquigarrow v \rightsquigarrow t_i$ ✓

Daniel Marx

Igor Razgon

Shadow Removal [STOC 2011]

Daniel Marx

Igor Razgon

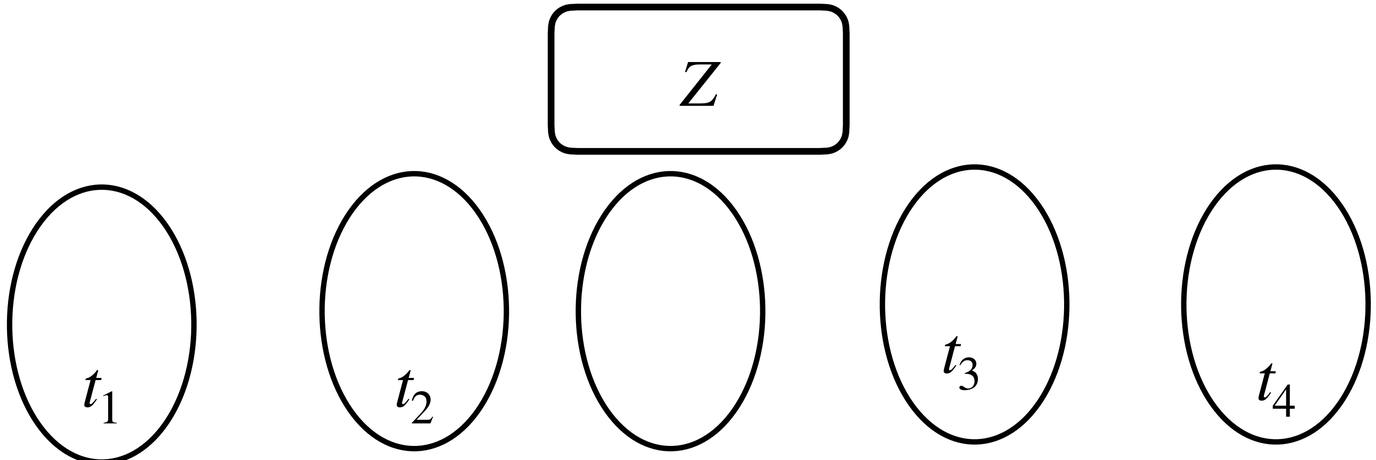
Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$

Shadowless solution w.r.t T:
$Z \subseteq V(G)$ is shadowless if
$\forall v \in V(G) \backslash Z$,
$\exists$ paths $T \rightsquigarrow v \rightsquigarrow T$

$t_i \rightsquigarrow v \rightsquigarrow t_j, i \neq j$

$t_i \rightsquigarrow v \rightsquigarrow t_i$



Shadow Removal [STOC 2011]

Daniel Marx

Igor Razgon

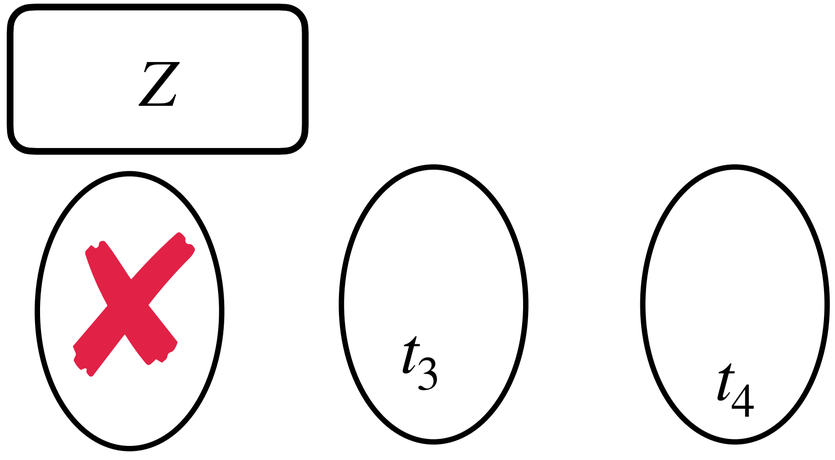
Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$

Shadowless solution w.r.t T:
$Z \subseteq V(G)$ is shadowless if
$\forall v \in V(G) \setminus Z$,
$\exists$ paths $T \rightsquigarrow v \rightsquigarrow T$

$t_i \rightsquigarrow v \rightsquigarrow t_j, i \neq j$

$t_i \rightsquigarrow v \rightsquigarrow t_i$



Shadow Removal [STOC 2011]

Daniel Marx

Igor Razgon

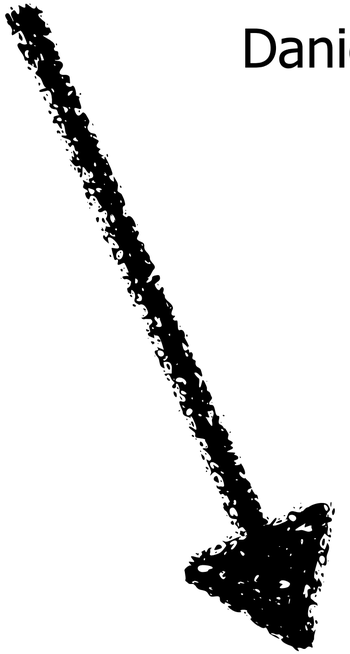
Eg: vertex-deletion **DIRECTED MULTIWAY CUT** (G, T, k)

$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$

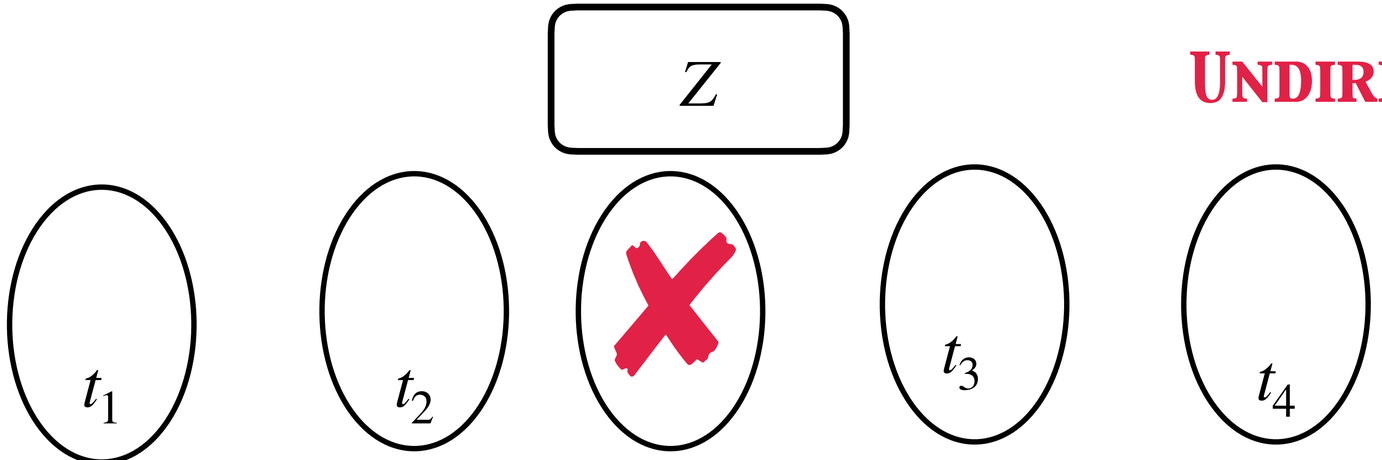
Shadowless solution w.r.t T:
$Z \subseteq V(G)$ is shadowless if
$\forall v \in V(G) \setminus Z$,
$\exists$ paths $T \rightsquigarrow v \rightsquigarrow T$

$t_i \rightsquigarrow v \rightsquigarrow t_j, i \neq j$ ✗

$t_i \rightsquigarrow v \rightsquigarrow t_i$ ✓



UNDIRECTED MULTIWAY CUT



Shadow Removal [STOC 2011]

~Random sampling of important cuts.

Daniel Marx

Igor Razgon

Inherently greedy: If there is a solution, assume WLOG that there is a **shadowless solution**.
Now the goal is to find a shadowless solution, which in many cases is easier.

DIRECTED MULTIWAY is FPT.

UNDIRECTED MULTICUT is FPT.

Shadow Removal [STOC 2011]

~Random sampling of important cuts.

Daniel Marx

Igor Razgon

Inherently greedy: If there is a solution, assume WLOG that there is a **shadowless solution**.
Now the goal is to find a shadowless solution, which in many cases is easier.

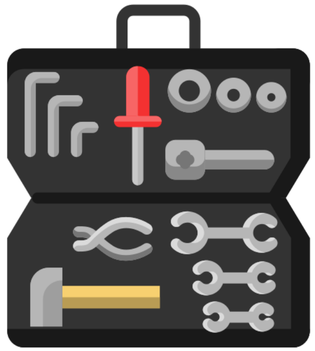
DIRECTED MULTIWAY is FPT.

UNDIRECTED MULTICUT is FPT.

Pre-pandemic~2016:Landscape

Important Cuts

Shadow Removal

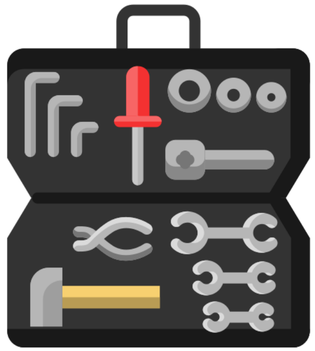


Other tools,
mostly for undirected problems

Pre-pandemic~2016:Landscape

Important Cuts

Shadow Removal



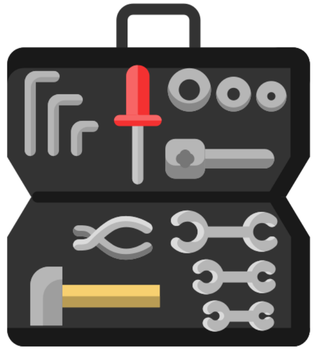
Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected			—	—
Directed				

Pre-pandemic~2016:Landscape

Important Cuts

Shadow Removal



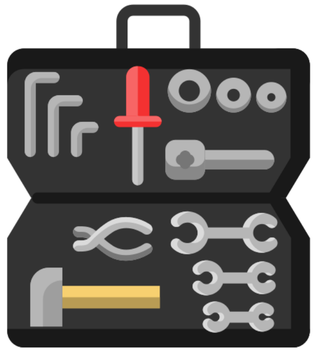
Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected			—	—
Directed				

Pre-pandemic~2016:Landscape

Important Cuts

Shadow Removal



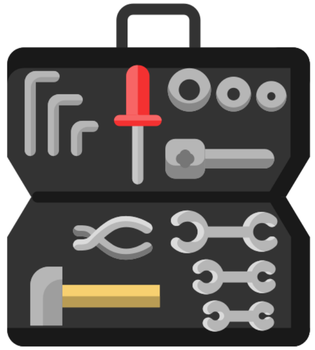
Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected			—	—
Directed				

Pre-pandemic~2016:Landscape

Important Cuts

Shadow Removal



Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected			—	—
Directed				

Pre-pandemic~2016: Landscape Contd.

DIRECTED MULTICUT

Pre-pandemic~2016: Landscape Contd.

DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

Pre-pandemic~2016: Landscape Contd.

DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

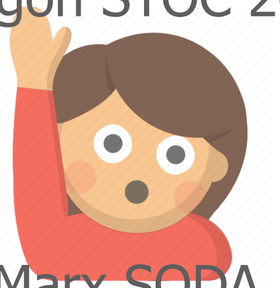
When p is constant?

Pre-pandemic~2016: Landscape Contd.

DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

When p is constant?



- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Pre-pandemic~2016: Landscape Contd.

DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

When p is constant?

- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Reduce to **DIRECTED MULTIWAY CUT** with 2 terminals

Pre-pandemic~2016: Landscape Contd.

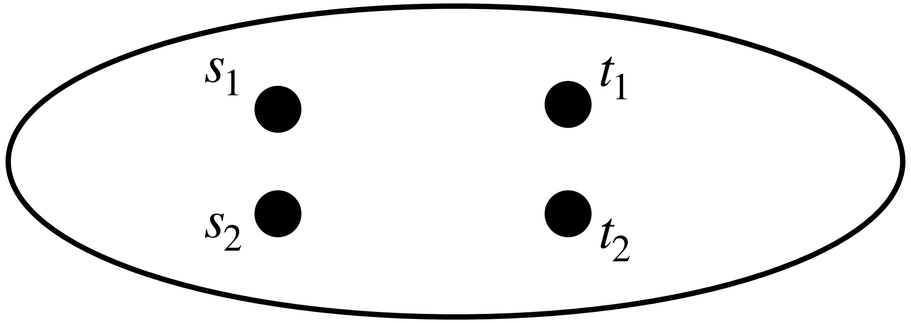
DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

When p is constant?

- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Reduce to **DIRECTED MULTIWAY CUT** with 2 terminals



Pre-pandemic~2016: Landscape Contd.

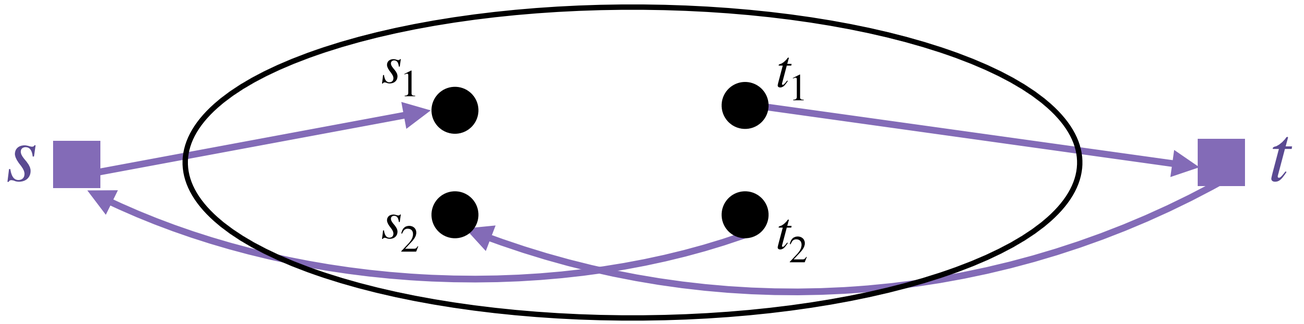
DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

When p is constant?

- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Reduce to **DIRECTED MULTIWAY CUT** with 2 terminals



Pre-pandemic~2016: Landscape Contd.

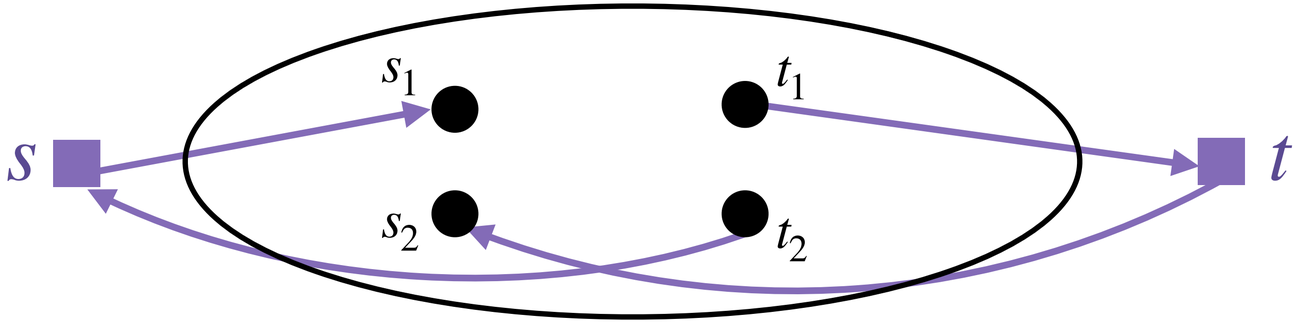
DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

When p is constant?

- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Reduce to **DIRECTED MULTIWAY CUT** with 2 terminals



- W[1]-hard even with 4 terminal pairs [Pilipczuk, Wahlström SODA 2016, ACM TOCT 2018]

Pre-pandemic~2016: Landscape Contd.

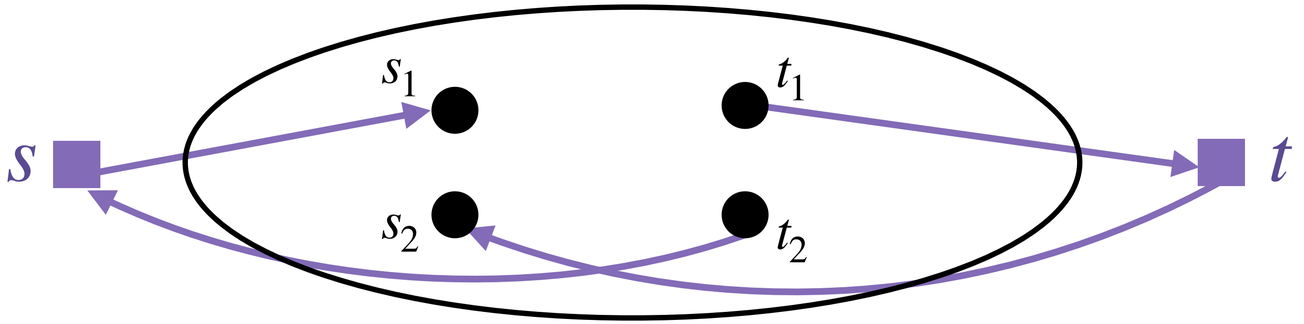
DIRECTED MULTICUT

- W[1]-hard (parameterized by $k+p$) [Marx, Razgon STOC 2011, SICOMP 2014]

When p is constant?

- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Reduce to **DIRECTED MULTIWAY CUT** with 2 terminals



- W[1]-hard even with 4 terminal pairs [Pilipczuk, Wahlström SODA 2016, ACM TOCT 2018]

3 terminal pairs

Pre-pandemic~2016: Open questions

I DIRECTED MULTICUT

Pre-pandemic~2016: Open questions

I **Directed Multicut**

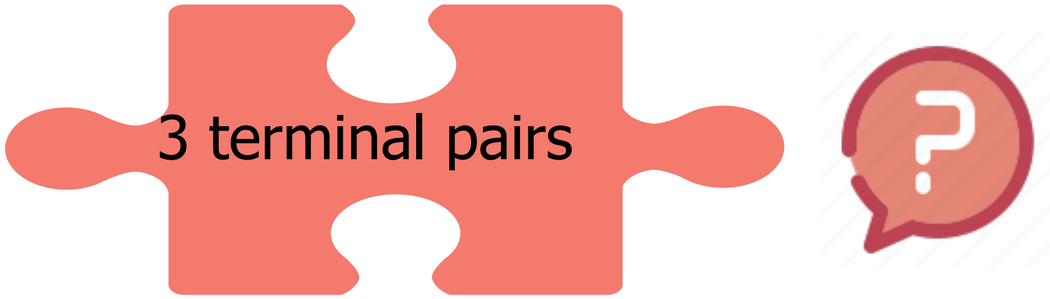
II

Weighted settings?

Find a solution of **size** at most k and **weight** at most W.
Parameter: k

Pre-pandemic~2016: Open questions

I **Directed Multicut**



II

Weighted settings?

Find a solution of **size** at most k and **weight** at most W.
Parameter: k

III

Boolean MinCSP dichotomy?

FPT v/s W[1]-hard dichotomy for Boolean MinCSP?

Known: Constant-factor FPT approximation classification
[Bonnet, Egri, Marx ESA 2016]

Cut problems of our interest

The greedy era (pre-pandemic)

- Tools:
  - Important Cuts
  - Shadow Removal
- PC Cut landscape
- Open Questions

Flow-augmentation (the pandemic)

- What is it?
- Easy Application:
  - Weighted s-t cut
- Other applications:
  - ℓ-Chain SAT

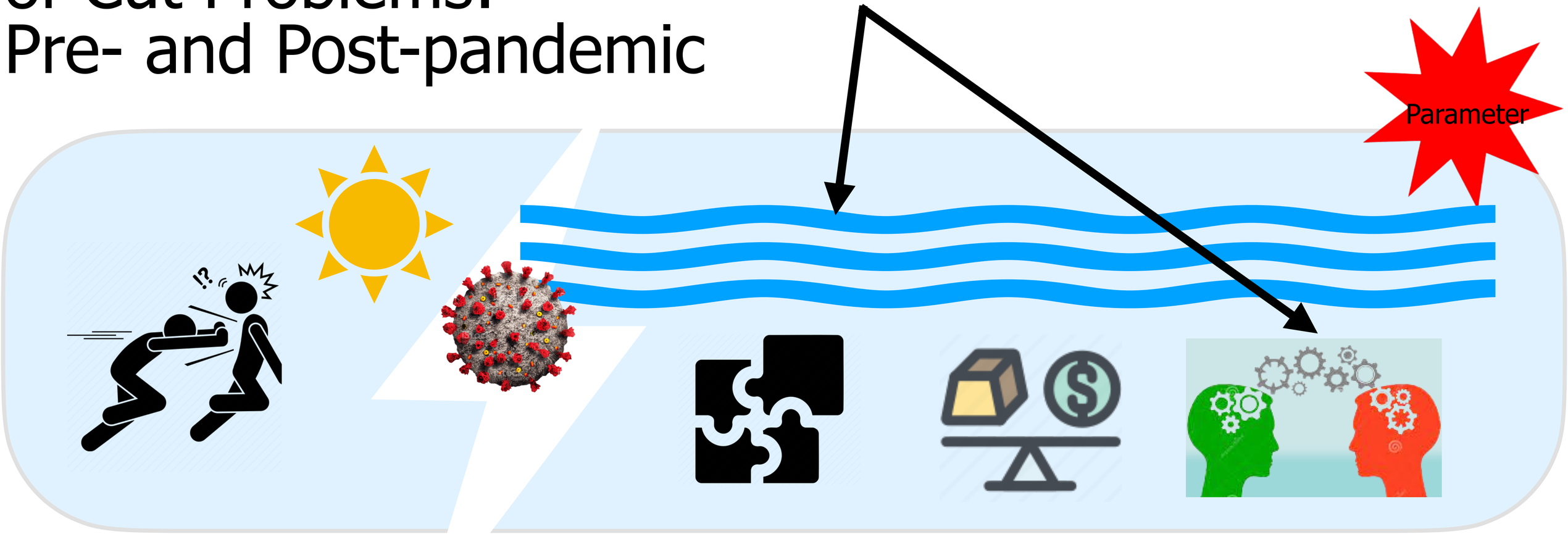
MinCSPs ("Post"-pandemic)

Boolean MinCSP dichotomy

Weighted Cut Problems ("Post"-pandemic)

- Weighted Edge Multiway Cut
- Weighted Edge Multicut
- Weighted Vertex Multicut
- Weighted (Subset) DFAS

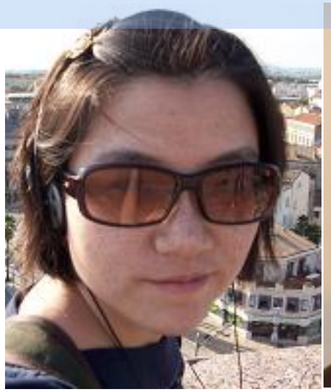
Parameterized Complexity Landscape of Cut Problems: Pre- and Post-pandemic



Greedy

Non-Greedy

Flow augmentation [STOC 2022]

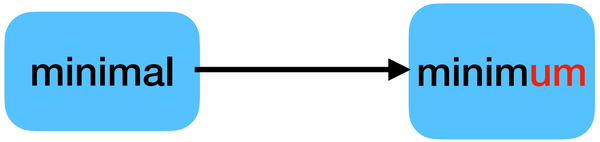


Eunjung Kim

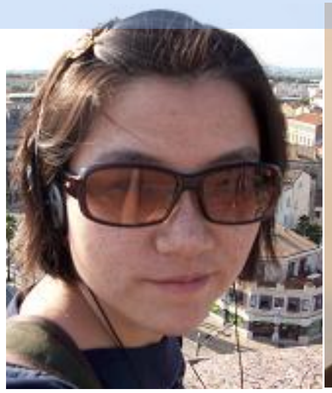
Stefan Krastch

Marcin Pilipczuk

Magnus Wahlström



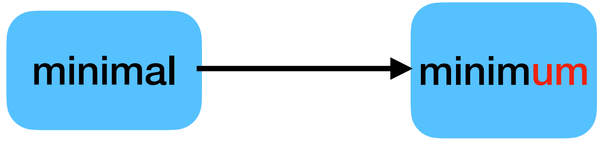
Flow augmentation [STOC 2022]


Eunjung Kim


Stefan Krastch

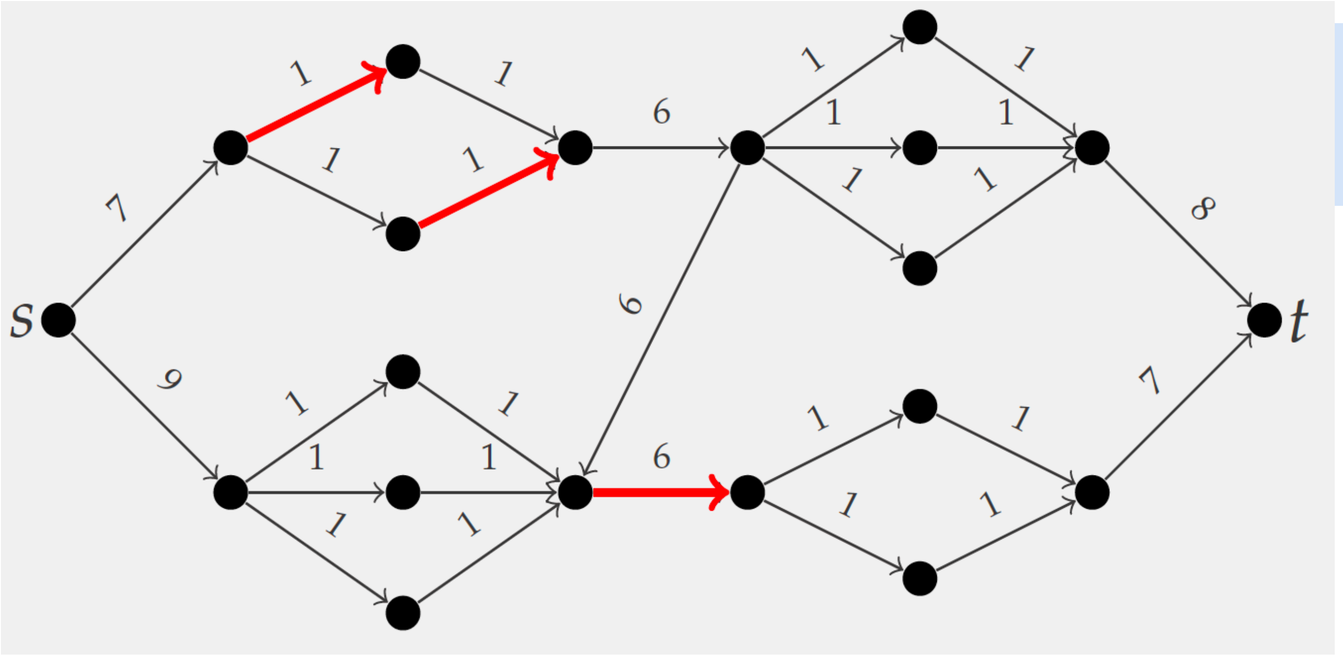

Marcin Pilipczuk


Magnus Wahlström



Input: Directed graph G, vertices s, t, integer k
Output: A supergraph G' of G obtained by adding new arcs A

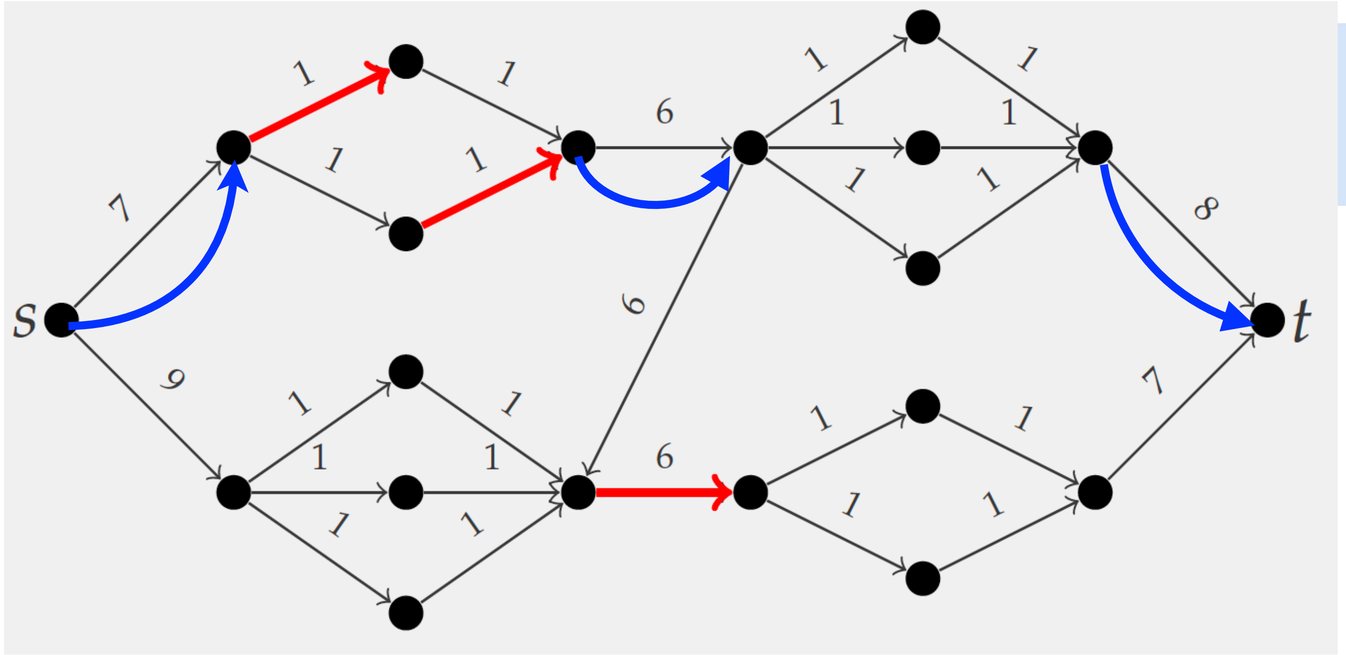
Guarantee: any **minimal** s-t separator Z of size at most k in G,
is a **minimum** s-t separator in G',
with probability $2^{-\mathcal{O}(k^4 \log k)}$



minimal → minimum

Input: Directed graph G, vertices s, t, integer k
Output: A supergraph G' of G obtained by adding new arcs A

Guarantee: any **minimal** s-t separator Z of size at most k in G,
is a **minimum** s-t separator in G',
with probability $2^{-\mathcal{O}(k^4 \log k)}$

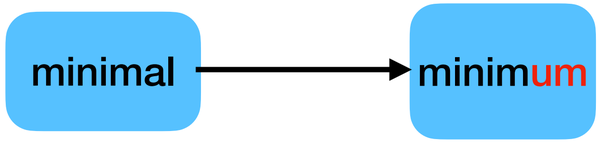


minimal → minimum

Input: Directed graph G, vertices s, t, integer k
Output: A supergraph G' of G obtained by adding new arcs A

Guarantee: any **minimal** s-t separator Z of size at most k in G,
is a **minimum** s-t separator in G',
with probability $2^{-\mathcal{O}(k^4 \log k)}$

Flow augmentation



Weighted s-t cut

Given a directed graph G, weight function w-old : $E(G) \to \mathbb{N}$, k, W
find an s-t cut Z in G of size at most k and weight at most W.

(NP-hard)

Flow augmentation

minimal → minimum

Weighted s-t cut

(NP-hard)

Given a directed graph G, weight function w-old : $E(G) \to \mathbb{N}$, k, W
find an s-t cut Z in G of size at most k and weight at most W.

Do flow-augmentation
and get G'.

Flow augmentation

minimal → minimum

Weighted s-t cut

(NP-hard)

Given a directed graph G, weight function w-old : $E(G) \to \mathbb{N}$, k, W
find an s-t cut Z in G of size at most k and weight at most W.

Do flow-augmentation
and get G'.

[Z is a minimum s-t cut
in the new graph.]

Flow augmentation

minimal → minimum

Weighted s-t cut

(NP-hard)

Given a directed graph G, weight function w-old : $E(G) \to \mathbb{N}$, k, W
find an s-t cut Z in G of size at most k and weight at most W.

Do flow-augmentation and get G'.

[Z is a minimum s-t cut in the new graph.]

Compute min s-t cut value, say λ, in G'.
If $\lambda > k$, report No.

Flow augmentation

minimal → minimum

Weighted s-t cut

(NP-hard)

Given a directed graph G, weight function w-old : $E(G) \to \mathbb{N}$, k, W
find an s-t cut Z in G of size at most k and weight at most W.

Do flow-augmentation and get G'.

[Z is a minimum s-t cut in the new graph.]

Compute min s-t cut value, say λ, in G'.
If $\lambda > k$, report No.

Assign new weights
w-new(e) = w-old$(e) + \tilde{W}$,
where $\tilde{W} = \sum_{e \in E(G)}$ w-old$(e) + 1$

Flow augmentation

minimal → minimum

Weighted s-t cut

(NP-hard)

Given a directed graph G, weight function w-old : $E(G) \to \mathbb{N}$, k, W
find an s-t cut Z in G of size at most k and weight at most W.

Do flow-augmentation and get G'.

[Z is a minimum s-t cut in the new graph.]

Compute min s-t cut value, say λ, in G'.
If $\lambda > k$, report No.

Assign new weights
w-new(e) = w-old(e) + $\tilde{W}$,
where $\tilde{W} = \sum_{e \in E(G)}$ w-old$(e) + 1$

Is there an s-t cut in G' whose w-new is at most $\lambda\tilde{\mathbf{W}} + \mathbf{W}$?

Constraint Satisfaction Problems (CSP)

Variables, Domain, Constraints

CSP is defined by: (1) the domain D set and (2) the set of allowed constraints Γ.

Example 1: 2-SAT

$D = \{0,1\}$

Γ= all 2-clauses

Instance:

$V = \{x_1, x_2, x_3\}$

Constraints: $(x_1 \vee x_2), (\neg x_2 \vee x_3), (\neg x_1 \vee \neg x_2)$

Constraint Satisfaction Problems (CSP)

Variables, Domain, Constraints

CSP is defined by: (1) the domain D set and (2) the set of allowed constraints Γ.

For a fixed CSP (D, Γ), an instance consists of :

- A set of variables
- A set of constraints from Γ applied to tuples of variables.

Goal: Find a satisfying assignment (from the domain to the variables that satisfies all constraints).

Example 1: 2-SAT

$D = \{0,1\}$

Γ= all 2-clauses

Instance:

$V = \{x_1, x_2, x_3\}$

Constraints: $(x_1 \vee x_2), (\neg x_2 \vee x_3), (\neg x_1 \vee \neg x_2)$

Constraint Satisfaction Problems (CSP)

Variables, Domain, Constraints

CSP is defined by: (1) the domain D set and (2) the set of allowed constraints Γ.

For a fixed CSP (D, Γ), an instance consists of :

- A set of variables
- A set of constraints from Γ applied to tuples of variables.

Goal: Find a satisfying assignment (from the domain to the variables that satisfies all constraints).

Example 1: 2-SAT

$D = \{0,1\}$

Γ= all 2-clauses

Instance:

$V = \{x_1, x_2, x_3\}$

Constraints: $(x_1 \vee x_2), (\neg x_2 \vee x_3), (\neg x_1 \vee \neg x_2)$

Example 2: 3-Coloring

$D = \{0,1,2\}$

Γ= $\{ \neq \}$

Instance: Graph G

$V = V(G)$

Constraints: for each $uv \in E(G), u \neq v$

Constraint Satisfaction Problems (CSP)

Variables, Domain, Constraints

CSP is defined by: (1) the domain D set and (2) the set of allowed constraints Γ.

For a fixed CSP (D, Γ), an instance consists of :

- A set of variables
- A set of constraints from Γ applied to tuples of variables.

Goal: Find a satisfying assignment (from the domain to the variables that satisfies all constraints).

Example 1: 2-SAT

Constraints: $(x_1 \vee x_2), (\neg x_2 \vee x_3), (\neg x_1 \vee \neg x_2)$

Example 2: 3-Coloring

Constraints: for each $uv \in E(G), u \neq v$

Theorem [Bulatov, Zhuk 2017]: For every finite D and Γ, the corresponding CSP is either polynomial-time solvable or NP-complete.

MinCSP

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

MinCSP

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

Min 2-SAT: captures a range of problems like **Edge Bipartization**, **Odd Cycle Transversal**, **Above-guarantee Vertex Cover**, **Köning Vertex Deletion** and **Split Vertex Deletion.**

MinCSP

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

Min 2-SAT: captures a range of problems like **Edge Bipartization**, **Odd Cycle Transversal**, **Above-guarantee Vertex Cover**, **Köning Vertex Deletion** and **Split Vertex Deletion**.

- Interesting if CSP(D, Γ) is polynomial-time.
- Trivial $n^{\mathcal{O}(k)}$ algorithm.
- Is it FPT parameterized by k?

MinCSP

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

Min 2-SAT: captures a range of problems like **Edge Bipartization**, **Odd Cycle Transversal**, **Above-guarantee Vertex Cover**, **Köning Vertex Deletion** and **Split Vertex Deletion**.

- Interesting if CSP(D, Γ) is polynomial-time.
- Trivial $n^{\mathcal{O}(k)}$ algorithm.
- Is it FPT parameterized by k?

Weighted MinCSP (D, Γ): Each constraint has a weight.

Can we delete at most k constraints of total weight W to make the resulting instance satisfiable?

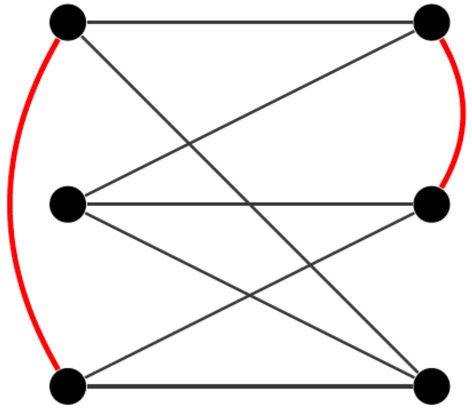
MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

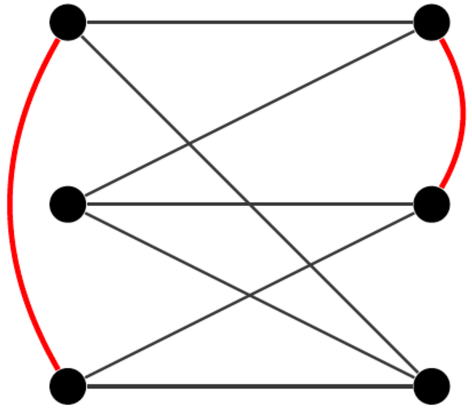
- $D = \{0,1\}, \Gamma = \{x \neq y\}$



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

EDGE BIPARTIZATION



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

EDGE BIPARTIZATION

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

EDGE BIPARTIZATION

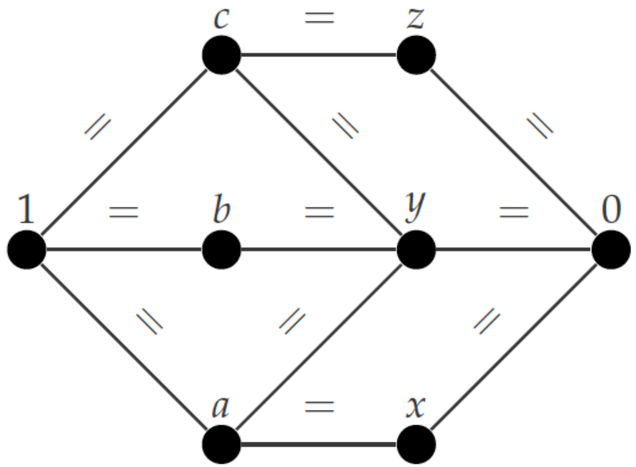
- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

EDGE BIPARTIZATION

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$



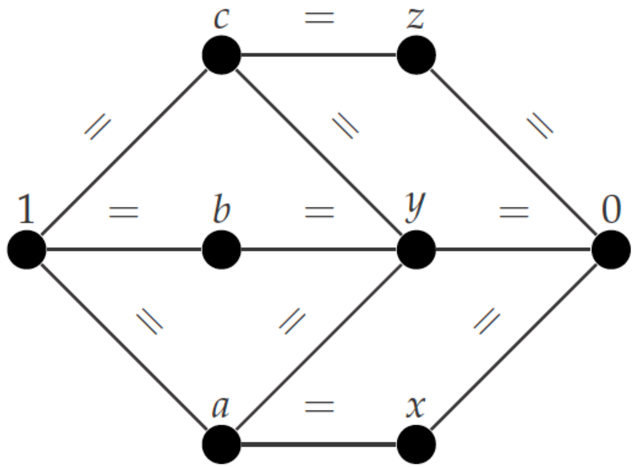
MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \rightarrow x, x \rightarrow 0, x \rightarrow y\}$

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

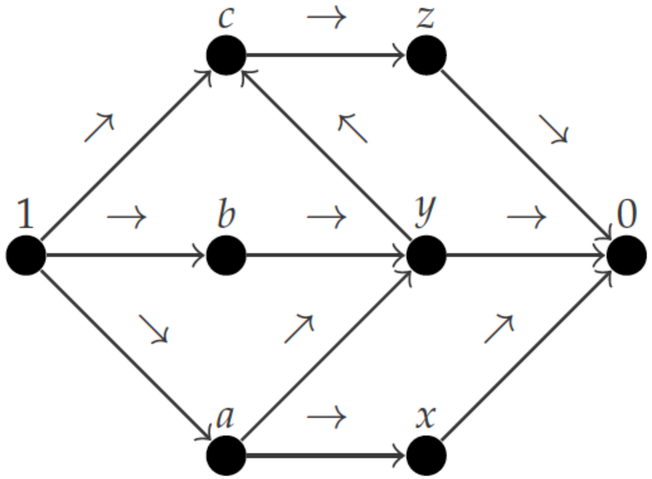
- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

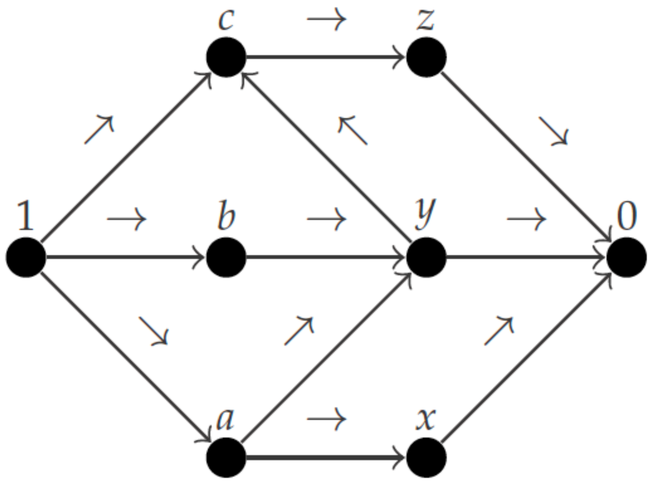
Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

Directed s-t cut



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \rightarrow x, x \rightarrow 0, x \rightarrow y\}$

Directed s-t cut

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \rightarrow x, x \rightarrow 0, x \rightarrow y\}$

Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \rightarrow x, x \rightarrow 0, (x \rightarrow y) \wedge (u \rightarrow v)\}$

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

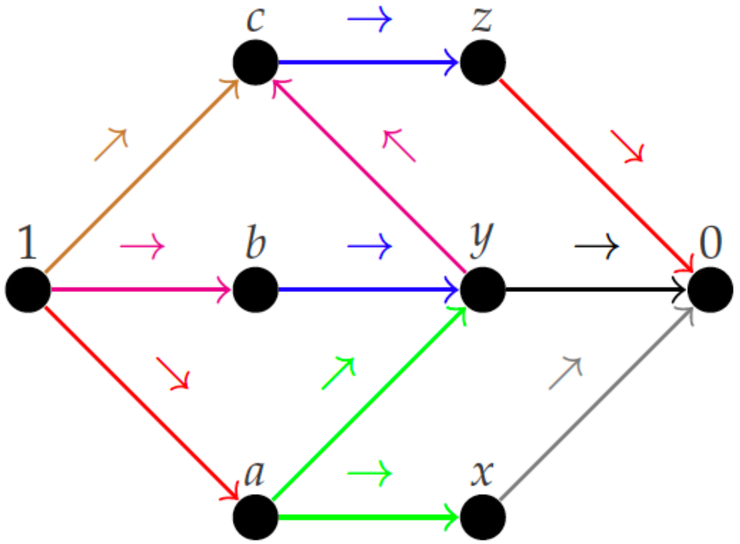
- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (u \to v)\}$



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

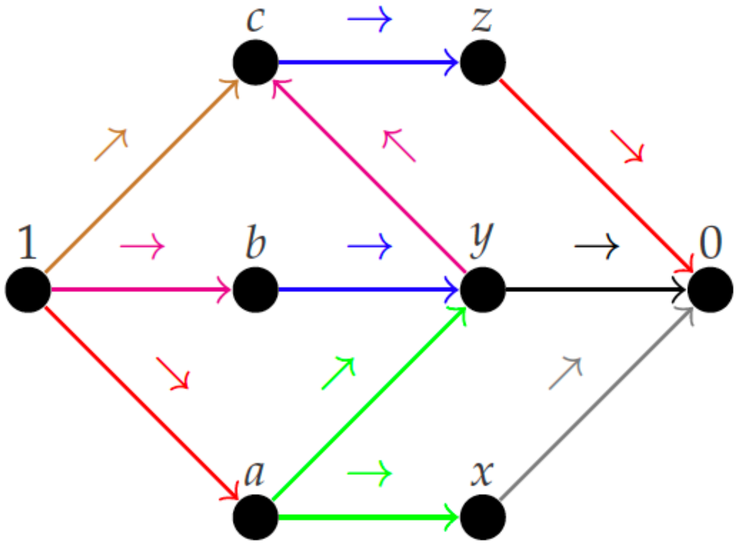
Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (u \to v)\}$

Bundled s-t cut W[1]-hard



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \rightarrow x, x \rightarrow 0, x \rightarrow y\}$

Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \rightarrow x, x \rightarrow 0, (x \rightarrow y) \wedge (u \rightarrow v)\}$

Bundled s-t cut W[1]-hard

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (u \to v)\}$

Bundled s-t cut W[1]-hard

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (y \to z) \wedge (z \to v)\}$

MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

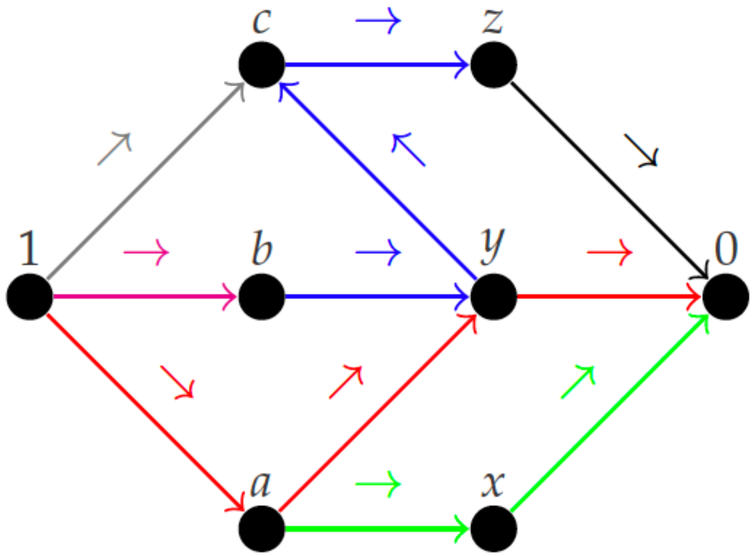
- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (u \to v)\}$

Bundled s-t cut W[1]-hard

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (y \to z) \wedge (z \to v)\}$



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

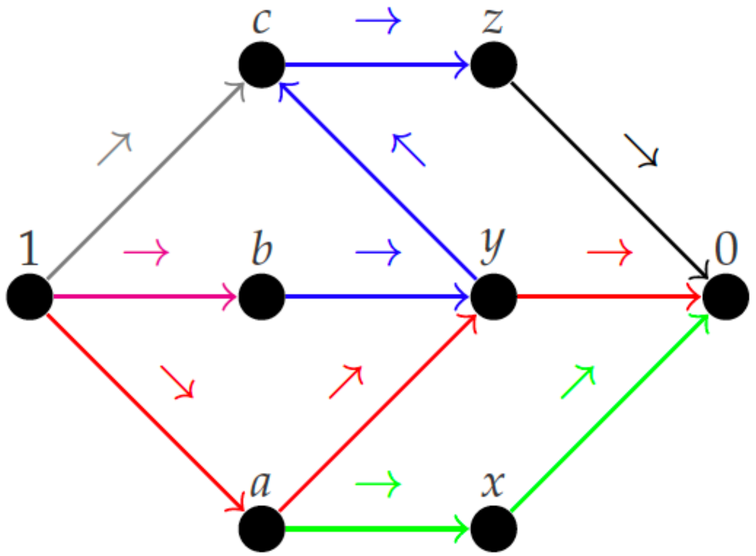
Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (u \to v)\}$

Bundled s-t cut W[1]-hard

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (y \to z) \wedge (z \to v)\}$

3-**Chain SAT** (ℓ-**Chain SAT**) FPT



MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

- $D = \{0,1\}, \Gamma = \{x \neq y\}$

Edge Bipartization

- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

Undirected s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

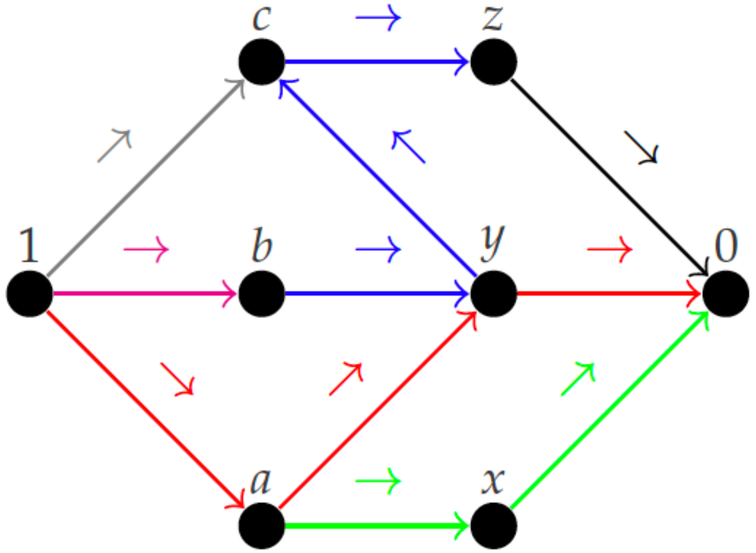
Directed s-t cut

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (u \to v)\}$

Bundled s-t cut W[1]-hard

- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, (x \to y) \wedge (y \to z) \wedge (z \to v)\}$

3-Chain SAT (ℓ-Chain SAT) FPT



Weighted Boolean MinCSP FPT v/s W[1]-hard dichotomy

Theorem [Kim, Kratsch, Pilipczuk, Wahlström SODA 2023]: FPT v/s W[1]-hard dichotomy for Weighted MinCSP with boolean domain.

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

Weighted Boolean MinCSP FPT v/s W[1]-hard dichotomy

Theorem [Kim, Kratsch, Pilipczuk, Wahlström SODA 2023]: FPT v/s W[1]-hard dichotomy for Weighted MinCSP with boolean domain.

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

Constraint graph (defined for every constraint)

- The vertex set is the variables in the constraint.
- A pair of variables v_i, v_j are **independent**, if every assignment to them can be extended to a satisfying assignment for this constraint.
- Put an edge between a pair of variables that is not independent.

FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

Constraint graph (defined for every constraint)

- The vertex set is the variables in the constraint.
- A pair of variables v_i, v_j are **independent**, if every assignment to them can be extended to a satisfying assignment for this constraint.
- Put an edge between a pair of variables that is not independent.

Eg: 4-CHAIN SAT

$$(v_1 \to v_2) \wedge (v_2 \to v_3) \wedge (v_3 \to v_4) \wedge (v_4 \to v_5)$$

FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

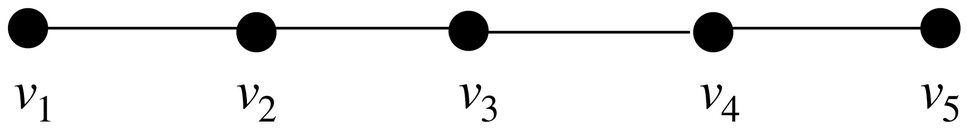
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

Constraint graph (defined for every constraint)

- The vertex set is the variables in the constraint.
- A pair of variables v_i, v_j are **independent,** if every assignment to them can be extended to a satisfying assignment for this constraint.
- Put an edge between a pair of variables that is not independent.

Eg: 4-CHAIN SAT

$$(v_1 \to v_2) \wedge (v_2 \to v_3) \wedge (v_3 \to v_4) \wedge (v_4 \to v_5)$$



FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

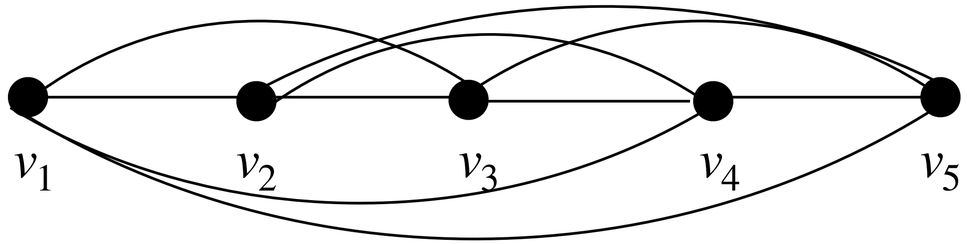
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

Constraint graph (defined for every constraint)

- The vertex set is the variables in the constraint.
- A pair of variables v_i, v_j are **independent,** if every assignment to them can be extended to a satisfying assignment for this constraint.
- Put an edge between a pair of variables that is not independent.

Eg: 4-CHAIN SAT

$$(v_1 \to v_2) \wedge (v_2 \to v_3) \wedge (v_3 \to v_4) \wedge (v_4 \to v_5)$$



FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

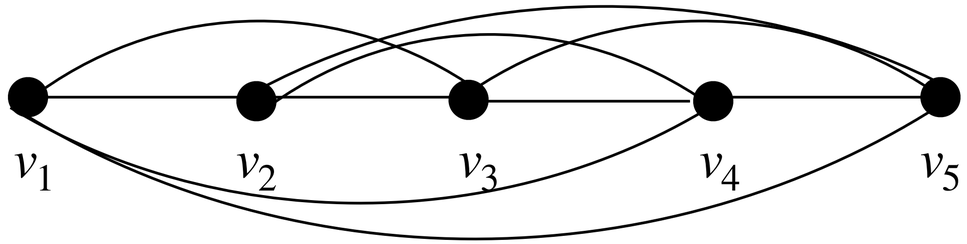
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

Constraint graph (defined for every constraint)

- The vertex set is the variables in the constraint.
- A pair of variables v_i, v_j are **independent,** if every assignment to them can be extended to a satisfying assignment for this constraint.
- Put an edge between a pair of variables that is not independent.

Eg: 4-CHAIN SAT

$$(v_1 \to v_2) \wedge (v_2 \to v_3) \wedge (v_3 \to v_4) \wedge (v_4 \to v_5)$$



FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

From ℓ-Chain SAT to Weighted Boolean MinCSP(Γ_{good}):
via **Weighted Digraph Pair Cut.**

Weighted Digraph Pair Cut

Directed graph G,
pairs of vertices (terminals) $(s_0, t_0), (s_1, t_1), \ldots, (s_p, t_p)$
weight function $\mathrm{wt} : E(G) \to \mathbb{N}$
positive integers k, W

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, $\mathrm{wt}(Z) \leq W$,
1. $G - Z$ has no $s_0 \to t_0$ path, and
2. for each $i \in \{1,\ldots,p\}$, either
$G - Z$ has no $s_0 \to s_i$ path or no $s_0 \to t_i$.

From ℓ-Chain SAT to Weighted Boolean MinCSP(Γ_{good}):
via **Weighted Digraph Pair Cut.**

Weighted Digraph Pair Cut

Directed graph G,
pairs of vertices (terminals) $(s_0, t_0), (s_1, t_1), \ldots, (s_p, t_p)$
weight function wt : $E(G) \to \mathbb{N}$
positive integers k, W

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, $\text{wt}(Z) \leq W$,
1. $G - Z$ has no $s_0 \to t_0$ path, and
2. for each $i \in \{1,\ldots,p\}$, either
$G - Z$ has no $s_0 \to s_i$ path or no $s_0 \to t_i$.

As Boolean MinCSP

$1 \to s_0$
$t_0 \to 0$
for each $uv \in E(G)$, $u \to v$
for each $(s_i, t_i), i \neq 0$, undeletable($\neg s_i \vee \neg t_i$)

Further remarks about Flow-augmentation

Further remarks about Flow-augmentation

- Flow-augmentation works for **star s-t cuts** (and not just minimal s-t cuts).

Further remarks about Flow-augmentation

- Flow-augmentation works for **star s-t cuts** (and not just minimal s-t cuts).
- Flow-augmentation ensures that the **core** of the star s-t cut becomes a **minimum s-t cut** (the core originally is a minimal s-t cut).

Further remarks about Flow-augmentation

- Flow-augmentation works for **star s-t cuts** (and not just minimal s-t cuts).
- Flow-augmentation ensures that the **core** of the star s-t cut becomes a **minimum s-t cut** (the core originally is a minimal s-t cut).
- Eg: Digraph Pair Cut solution is not necessarily a minimal s-t cut, but it is a star s-t cut.

Further remarks about Flow-augmentation

- Flow-augmentation works for **star s-t cuts** (and not just minimal s-t cuts).
- Flow-augmentation ensures that the **core** of the star s-t cut becomes a **minimum s-t cut** (the core originally is a minimal s-t cut).
- Eg: Digraph Pair Cut solution is not necessarily a minimal s-t cut, but it is a star s-t cut.
- Flow-augmentation outputs **witnessing flow** also (therefore FA is important even when core is a minimum s-t cut).

Further remarks about Flow-augmentation

- Flow-augmentation works for **star s-t cuts** (and not just minimal s-t cuts).
- Flow-augmentation ensures that the **core** of the star s-t cut becomes a **minimum s-t cut** (the core originally is a minimal s-t cut).
- Eg: Digraph Pair Cut solution is not necessarily a minimal s-t cut, but it is a star s-t cut.
- Flow-augmentation outputs **witnessing flow** also (therefore FA is important even when core is a minimum s-t cut).
- Undirected Flow-augmentation has a better success probability of $2^{-\mathcal{O}(k \log k)}$.

Further remarks about the dichotomy

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

Further remarks about the dichotomy

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

FPT island 2: Unweighted Boolean MinCSP(Γ) (when weighted is W[1]-hard) is FPT when:

Further remarks about the dichotomy

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

- **FPT island 2**: Unweighted Boolean MinCSP(Γ) (when weighted is W[1]-hard) is FPT when:
  - Γ is finite,

Further remarks about the dichotomy

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

FPT island 2: Unweighted Boolean MinCSP(Γ) (when weighted is W[1]-hard) is FPT when:

- Γ is finite,
- Γ is a $\wedge$ of negative clauses $(\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_r)$, positive 1-clauses (x) and implications $(x \rightarrow y)$, and

Further remarks about the dichotomy

Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

FPT island 2: Unweighted Boolean MinCSP(Γ) (when weighted is W[1]-hard) is FPT when:

- Γ is finite,
- Γ is a $\wedge$ of negative clauses $(\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_r)$, positive 1-clauses (x) and implications $(x \to y)$, and
- For every constraint in Γ, its arrow graph is $2K_2$-free.

Further remarks about the dichotomy

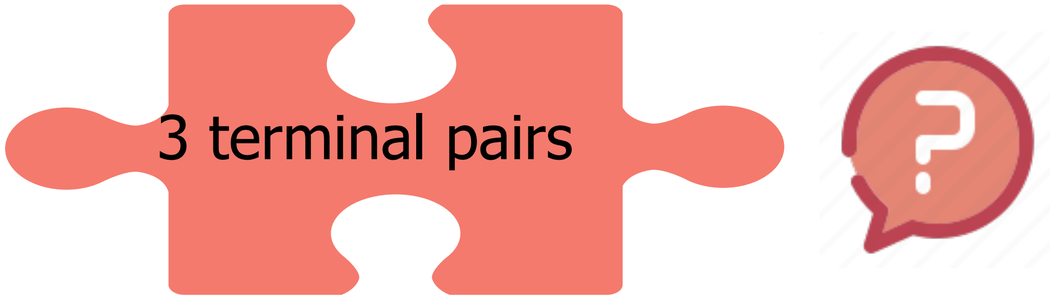
Either Weighted MinCSP(Γ) is FPT,
or Weighted MinCSP(Γ) is W[1]-hard, but MinCSP(Γ) is FPT,
or MinCSP(Γ) is W[1]-hard.

FPT island 2: Unweighted Boolean MinCSP(Γ) (when weighted is W[1]-hard) is FPT when:

- Γ is finite,
- Γ is a $\wedge$ of negative clauses $(\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_r)$, positive 1-clauses (x) and implications $(x \to y)$, and
- For every constraint in Γ, its arrow graph is $2K_2$-free.

Pre-pandemic~2016: Open questions

I **Directed Multicut**



II

Weighted settings?

Find a solution of **size** at most k and **weight** at most W.
Parameter: k

III

Boolean MinCSP dichotomy?

FPT v/s W[1]-hard dichotomy for Boolean MinCSP?

Known: Constant-factor FPT approximation classification
[Bonnet, Egri, Marx ESA 2016]

Pre-pandemic~2016: Open questions

I **Directed Multicut**

II

Weighted settings?

Find a solution of **size** at most k and **weight** at most W.
Parameter: k

III

FPT v/s W[1]-hard dichotomy for Boolean MinCSP?

Known: constant factor FPT approximation classfication
[Bonnet, Egri, Marx ESA 2016]

Pre-pandemic~2016: Open questions

I **Directed Multicut** 

FPT

[Heike, Jaffke, Lima, Masaryk, Pilipczuk, **Sharma**, Sorge SODA 2023]

Flow augmentation [STOC 2022]

Twin-width [FOCS 2020]

[Bonnet, Kim, Thomassé, Watrigant]

Generalization of the class of co-graphs

Cut problems of our interest

The greedy era (pre-pandemic)

- Tools:
  - Important Cuts
  - Shadow Removal
- PC Cut landscape
- Open Questions

Flow-augmentation (the pandemic)

- What is it?
- Easy Application:
  - Weighted s-t cut
- Other applications:
  - ℓ-Chain SAT

MinCSPs ("Post"-pandemic)

Boolean MinCSP dichotomy

Weighted Cut Problems ("Post"-pandemic)

- Weighted Edge Multiway Cut
- Weighted Edge Multicut
- Weighted Vertex Multicut
- Weighted (Subset) DFAS

FPT island 1: Weighted Boolean MinCSP(Γ_{good})

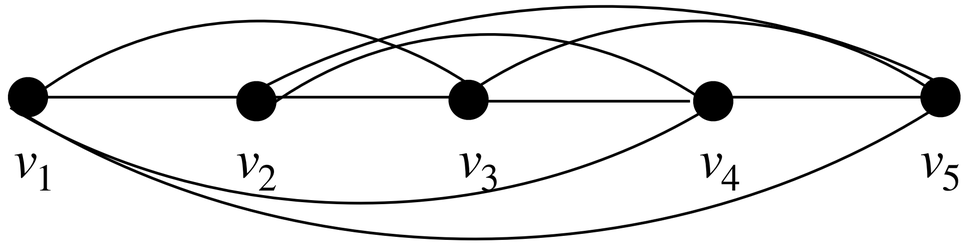
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.

Constraint graph (defined for every constraint)

- The vertex set is the variables in the constraint.
- A pair of variables v_i, v_j are **independent,** if every assignment to them can be extended to a satisfying assignment for this constraint.
- Put an edge between a pair of variables that is not independent.

Eg: 4-CHAIN SAT

$$(v_1 \to v_2) \wedge (v_2 \to v_3) \wedge (v_3 \to v_4) \wedge (v_4 \to v_5)$$



FPT Island: Weighted Boolean MinCSP(Γ_{good}) is FPT parameterized by k and the maximum arity over all constraints.

Weighted settings?

[Kim, Masaryk, Pilipczuk, **Sharma**, Wahlström, to appear SIDMA 2023]

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected	 MinCSP (=) (non-boolean)	 [KMP**S**W] MinCSP (= , ≠) (non-boolean)	—	—
Directed	W[1]-hard even with 2 terminals [HJLMP**S**S, SODA 2023]	W[1]-hard even with 2 pairs [HJLMP**S**S, SODA 2023]		 [KMP**S**W] MinCSP (< , ≤) (non-boolean)

UNDIRECTED MULTIWAYCUT

Undirected graph G,
vertices (terminals) $t_1, \ldots, t_p$
weight function wt : $E(G) \to \mathbb{N}$
positive integers k, W

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, $\text{wt}(Z) \leq W$ and
$G - Z$ has no $t_i - t_j$ path
for any $i, j \in \{1,\ldots,p\}$, $i \neq j$.

UNDIRECTED MULTIWAY CUT

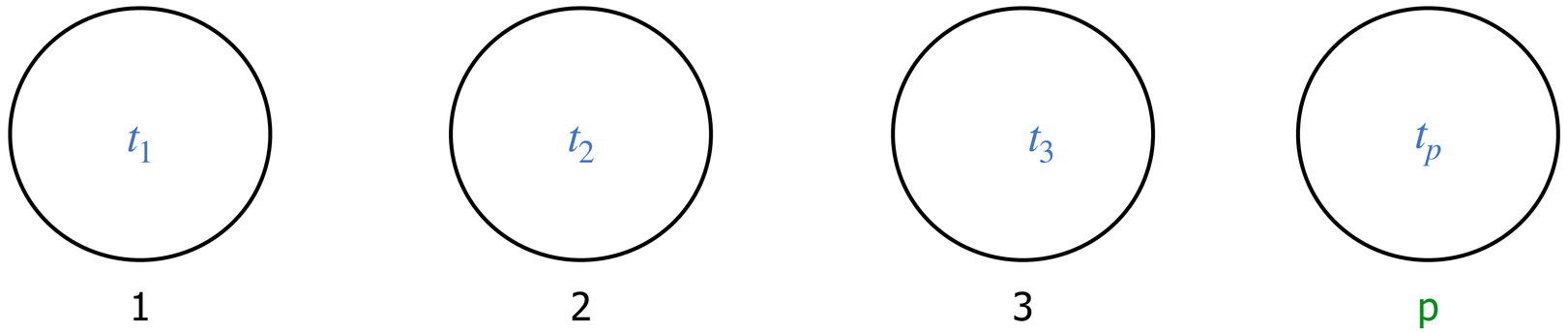
$\Longrightarrow$

MinCSP (=)

$T = \{t_1, \ldots, t_p\}$

Assume G is connected. Then $p \leq k + 1$.

G-Z (Z is a minimal solution)



connected components of G-Z

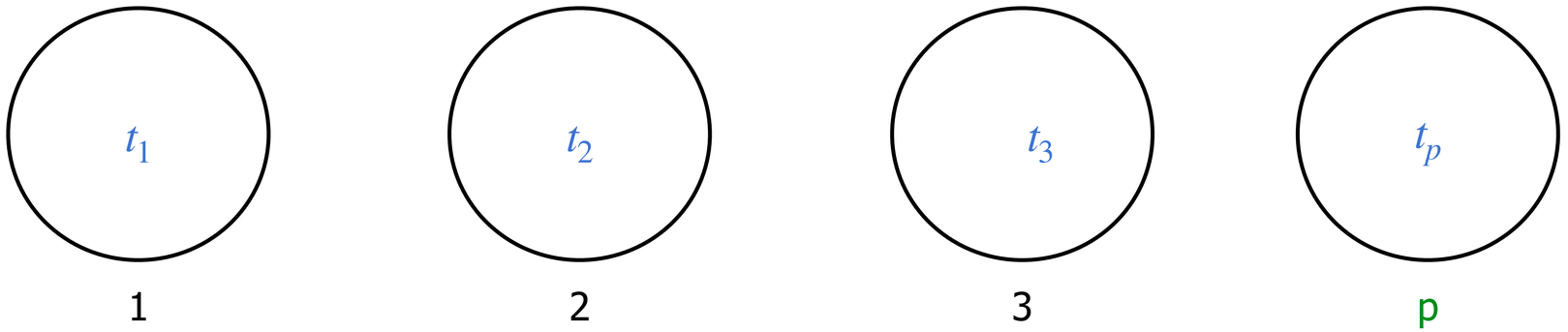
Undirected Multiway Cut

$\Longrightarrow$ MinCSP (=)

$T = \{t_1, \ldots, t_p\}$

Assume G is connected. Then $p \leq k + 1$.

G-Z (Z is a minimal solution)



connected components of G-Z

For every vertex $v \in V(G)$,	create a variable v	domain is $1,\ldots,p$
For every edge $uv \in E(G)$,	create a constraint $\mathbf{u = v}$	weight=wt(uv)
For every $t_i \in T$,	create an undeletable constraint $\mathbf{t_i = i}$	

Encode this

Domain	$\{1,\ldots,p\}$
$uv \in E(G)$	$\mathbf{u = v}$
$t_i \in T$	$\mathbf{t_i = i}$

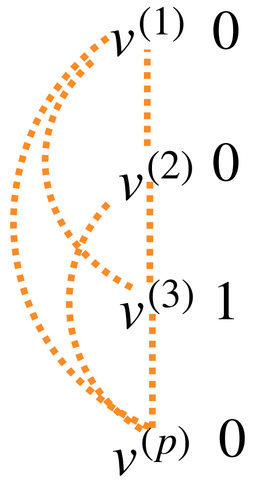
as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in Boolean

- Create **p** **variables** for every vertex v.
- Ensure that if v belongs to bucket, say 3, in G-Z, then $v^{(3)}$ is set to 1 and others are set to 0.
- If it belongs to bucket 0, then everything is assigned to 0.

- This can be ensured by adding undeletable constraints as shown on right.

$$\neg v^{(i)} \vee \neg v^{(j)}$$
$$\forall 1 \leq i < j \leq p$$



Encode this

Domain	$\{1,\ldots,p\}$
$uv \in E(G)$	$\mathbf{u = v}$
$t_i \in T$	$\mathbf{t_i = i}$

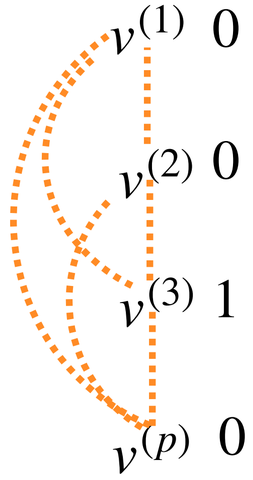
as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in Boolean

- Create **p variables** for every vertex v.
- Ensure that if v belongs to bucket, say 3, in G-Z, then $v^{(3)}$ is set to 1 and others are set to 0.
- If it belongs to bucket 0, then everything is assigned to 0.

- This can be ensured by adding undeletable constraints as shown on right.

$$\neg v^{(i)} \vee \neg v^{(j)}$$
$$\forall 1 \leq i < j \leq p$$



The unique superscript that gets assigned 1, tells the bucket this vertex will go into. If none of them gets assigned 1, then it goes to bucket 0.

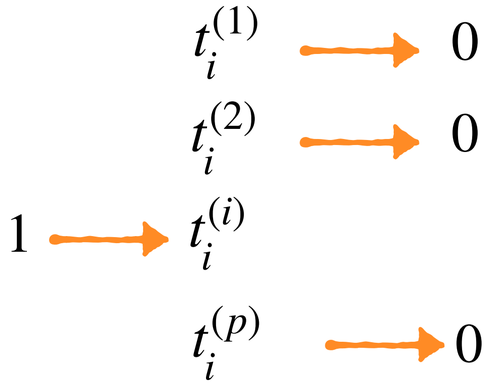
Encode this

Domain	$\{1,\ldots,p\}$
$uv \in E(G)$	$\mathbf{u} = \mathbf{v}$
$t_i \in T$	$\mathbf{t_i} = \mathbf{i}$

as MinCSP(Γ_{good})

Forcing the vertices of T in the correct bucket

- Force t_i in bucket i.
- This is done by adding undeletable constraints as shown on right.



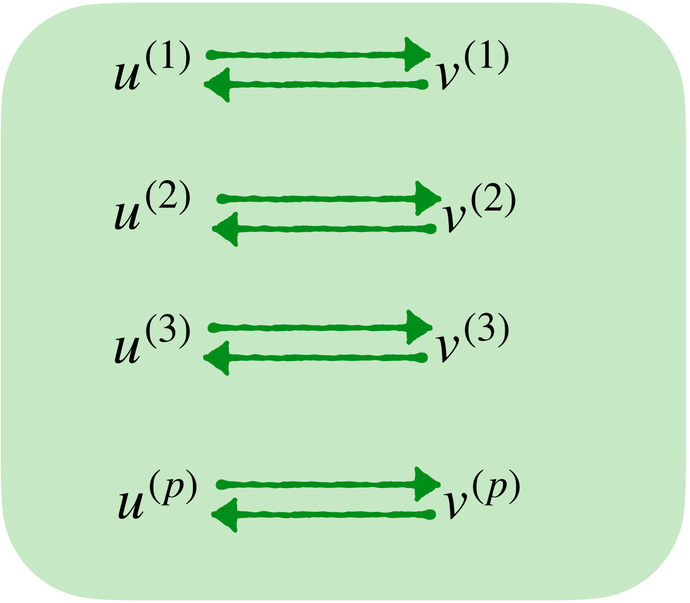
Encode this

Domain	$\{1,\ldots,p\}$
$uv \in E(G)$	$\mathbf{u = v}$
$t_i \in T$	$\mathbf{t_i = i}$

as MinCSP(Γ_{good})

Forcing an edge uv into the same bucket

- This is done by adding a constraint that is an **AND of all the clauses** shown on right.
- Weight of this constraint=$w(uv)$

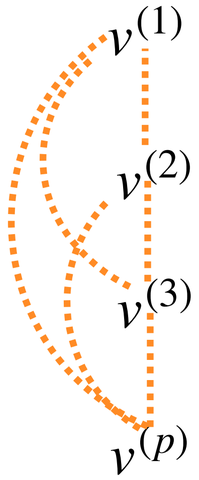


Encode this

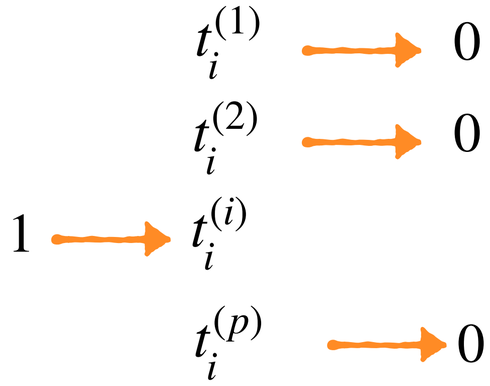
Domain	$\{1,\ldots,p\}$
$uv \in E(G)$	$\mathbf{u = v}$
$t_i \in T$	$\mathbf{t_i = i}$

as MinCSP(Γ_{good})

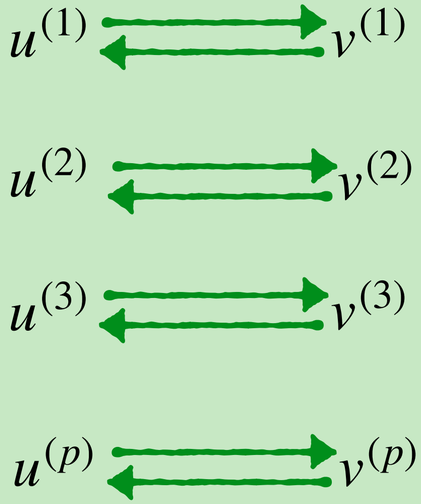
Encoding bucket numbers (domain) in Boolean



Forcing the vertices of T in the correct bucket



Forcing an edge uv in the same bucket

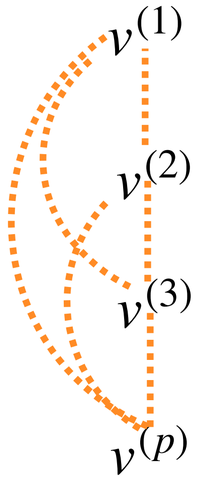


Encode this

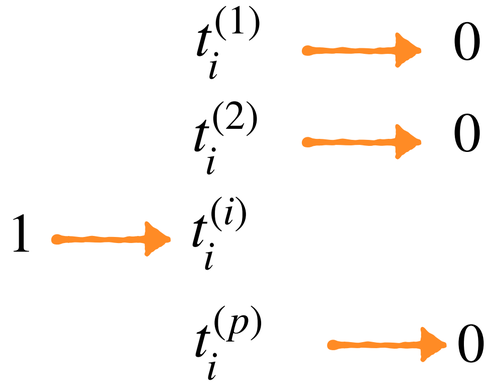
Domain	$\{1,\ldots,p\}$
$uv \in E(G)$	$\mathbf{u = v}$
$t_i \in T$	$\mathbf{t_i = i}$

as MinCSP(Γ_{good})

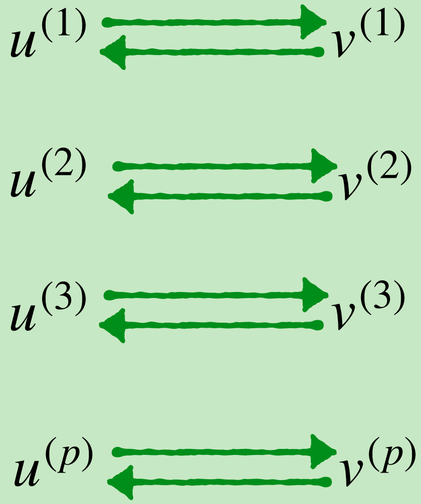
Encoding bucket numbers (domain) in Boolean



Forcing the vertices of T in the correct bucket



Forcing an edge uv in the same bucket



Encode this

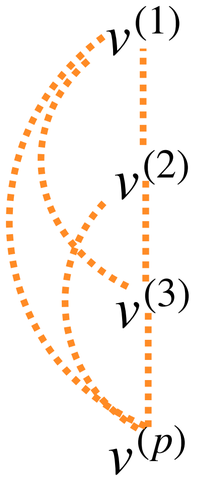
Domain $\{1,\ldots,p\}$

$uv \in E(G)$ $\quad \mathbf{u = v}$

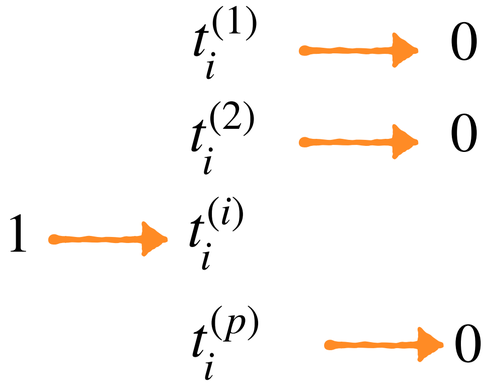
$t_i \in T$ $\quad \mathbf{t_i = i}$

as MinCSP(Γ_{good})

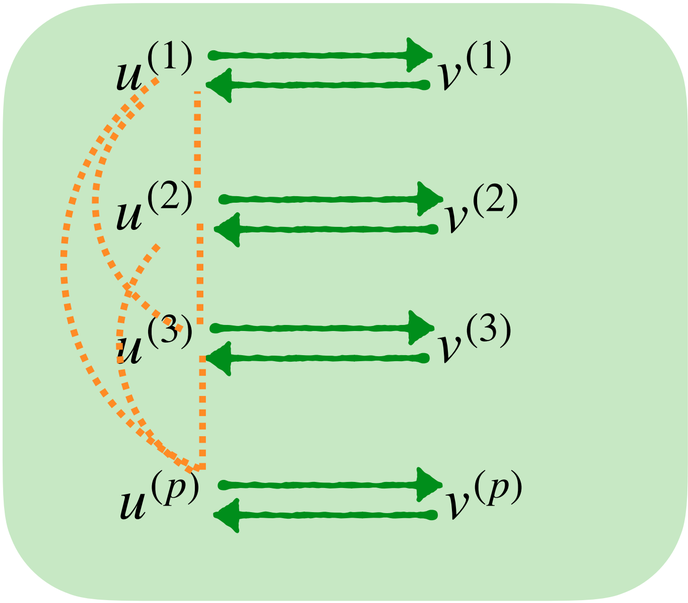
Encoding bucket numbers (domain) in Boolean



Forcing the vertices of T in the correct bucket



Forcing an edge uv in the same bucket



Undirected Edge Multicut

Undirected graph G,
pairs of vertices (terminals) $T = \{(s_1, t_1), \ldots, (s_p, t_p)\}$,
weight function $\text{wt} : E(G) \to \mathbb{N}$,
positive integers k, W

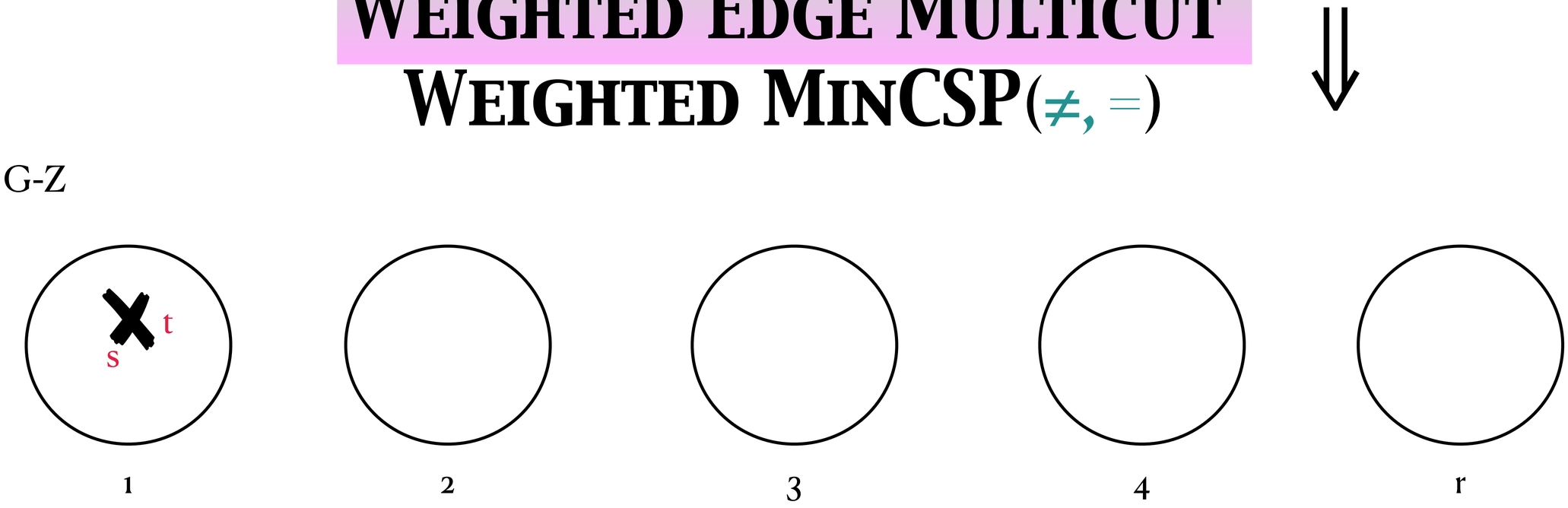
$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, $\text{wt}(Z) \leq W$ and
for each $i \in \{1,\ldots,p\}$,
$G - Z$ has no $s_i - t_i$ path.

$\equiv$

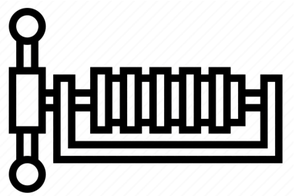
for any (s_i, t_i), s_i and t_i are in different connected components of $G - Z$.

Weighted Edge Multicut

Weighted MinCSP($\neq$, $=$)



For every vertex $v \in V(G)$,	create a variable v	domain is 1,....,r
For every edge $uv \in E(G)$,	create a constraint **u=v**	weight=wt(uv)
For every terminal pair $(s,t) \in T$,	create an undeletable constraint **s≠t**	



Weighted Edge Multicut

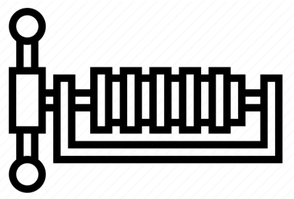
with Iterative Compression

Input++: $X \subseteq V(G)$, $|X| \leq k+1$, every terminal pair path intersects X.

Assume WLOG: $X=\{x_1,\ldots,x_{|X|}\}$ and X is an independent set.

n

$2^{\mathcal{O}(k \log k)}$



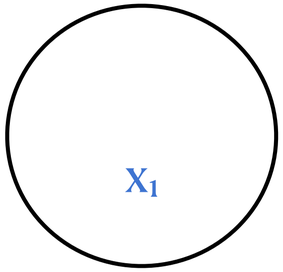
Weighted Edge Multicut

with Iterative Compression

G-Z

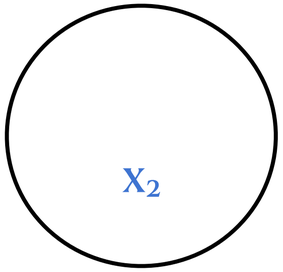
0

connected components
that do not contain X



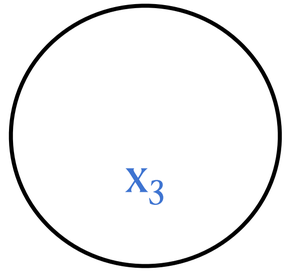
1

contains x_1



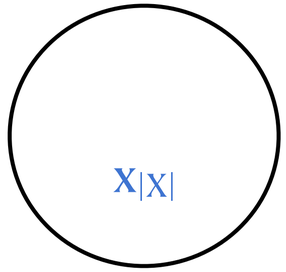
2

contains x_2



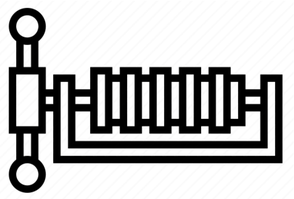
3

contains x_3



|X|

contains $x_{|X|}$



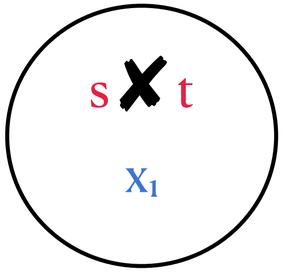
Weighted Edge Multicut

with Iterative Compression

G-Z

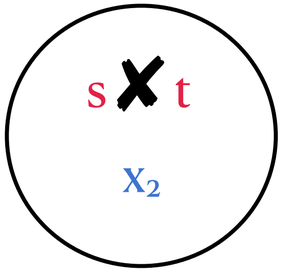
0

connected components
that do not contain X



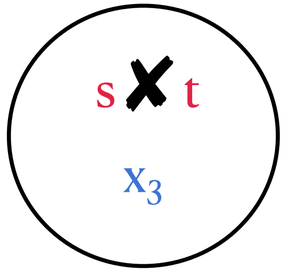
1

contains x_1



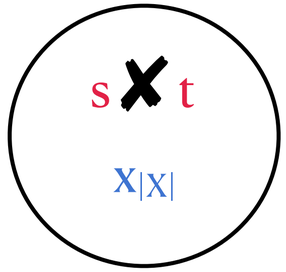
2

contains x_2



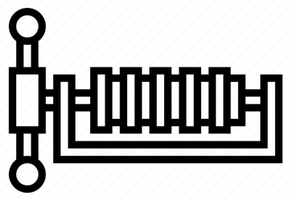
3

contains x_3



|X|

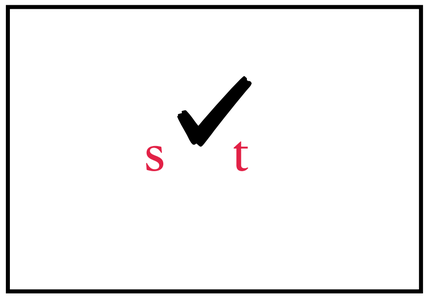
contains $x_{|X|}$



Weighted Edge Multicut

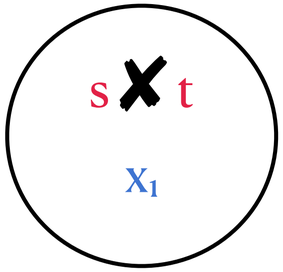
with Iterative Compression

G-Z



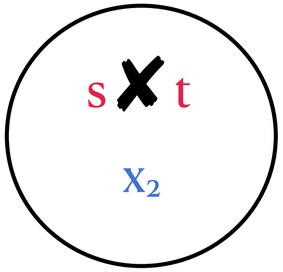
0

connected components
that do not contain X



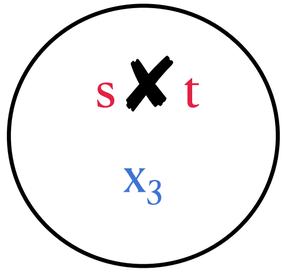
1

contains x_1



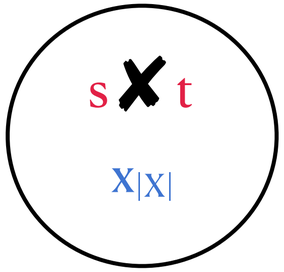
2

contains x_2



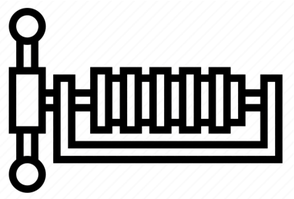
3

contains x_3



|X|

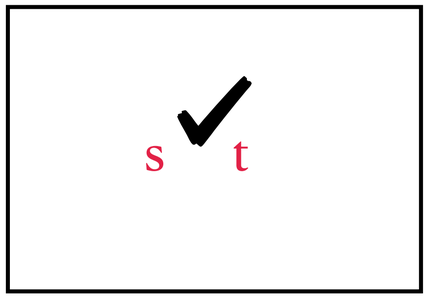
contains $x_{|X|}$



Weighted Edge Multicut

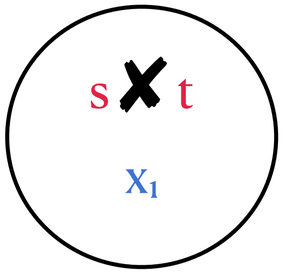
with Iterative Compression

G-Z



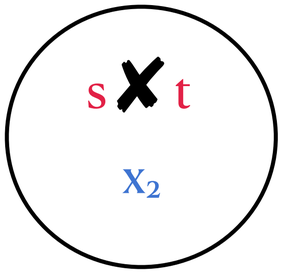
0

connected components
that do not contain X



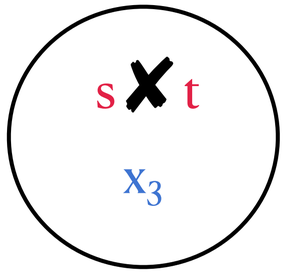
1

contains x_1



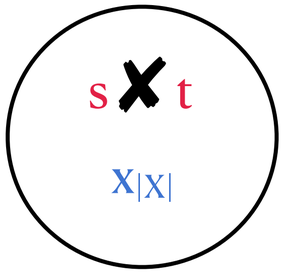
2

contains x_2



3

contains x_3

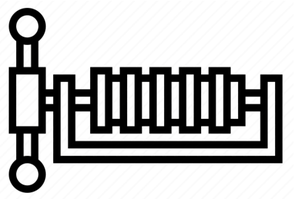


|X|

contains $x_{|X|}$

An **edge** can be inside a circle or inside the rectangle.

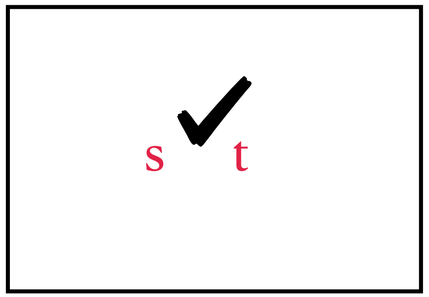
The endpoints of any terminal pair **can be inside a rectangle**, or in different circles (but not inside a circle).



Weighted Edge Multicut

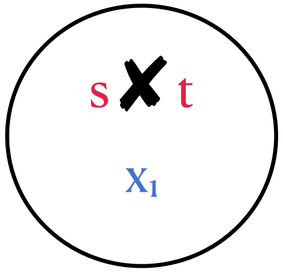
with Iterative Compression

G-Z



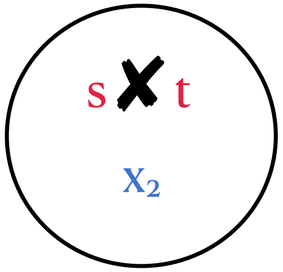
0

connected components that do not contain X



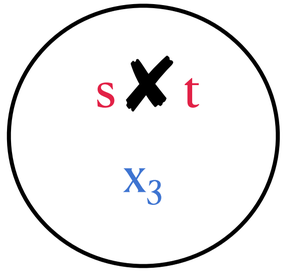
1

contains x_1



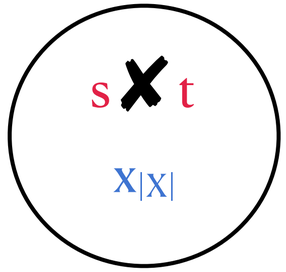
2

contains x_2



3

contains x_3



|X|

contains $x_{|X|}$

An **edge** can be inside a circle or inside the rectangle.

The endpoints of any terminal pair **can be inside a rectangle**, or in different circles (but not inside a circle).

Domain {0,1, …,|X|}

$uv \in E(G)$	**u=v**
$st \in T$	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

Encode this

Domain {0,1, …,|X|}	
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in Boolean

- Create **|X| variables** for every vertex v.
- Ensure that if v belongs to bucket, say 3, in G-Z, then $v^{(3)}$ is set to 1 and others are set to 0.
- If it belongs to bucket 0, then everything is assigned to 0.

$v^{(1)}$ 0

$v^{(2)}$ 0

$v^{(3)}$ 1

$v^{(|X|)}$ 0

Encode this

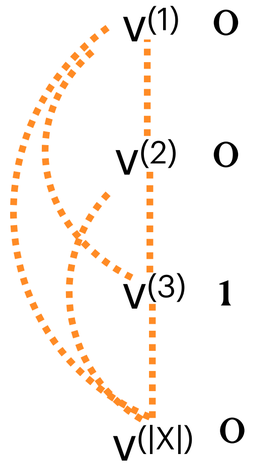
Domain {0,1, …,|X|}	
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in Boolean

- Create **|X| variables** for every vertex v.
- Ensure that if v belongs to bucket, say 3, in G-Z, then $v^{(3)}$ is set to 1 and others are set to 0.
- If it belongs to bucket 0, then everything is assigned to 0.
- This can be ensured by adding undeletable constraints as shown on right.

$$\neg v^{(i)} \vee \neg v^{(j)}$$
$$\forall 1 \leq i < j \leq |X|$$



Encode this

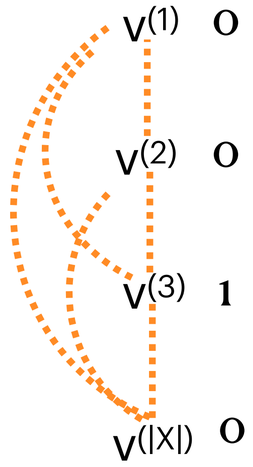
Domain {0,1, …,|X|}	
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in Boolean

- Create **|X| variables** for every vertex v.
- Ensure that if v belongs to bucket, say 3, in G-Z, then $v^{(3)}$ is set to 1 and others are set to 0.
- If it belongs to bucket 0, then everything is assigned to 0.
- This can be ensured by adding undeletable constraints as shown on right.

$$\neg v^{(i)} \vee \neg v^{(j)}$$
$$\forall 1 \leq i < j \leq |X|$$



The unique superscript that gets assigned 1, tells the bucket this vertex will go into. If none of them gets assigned 1, then it goes to bucket 0.

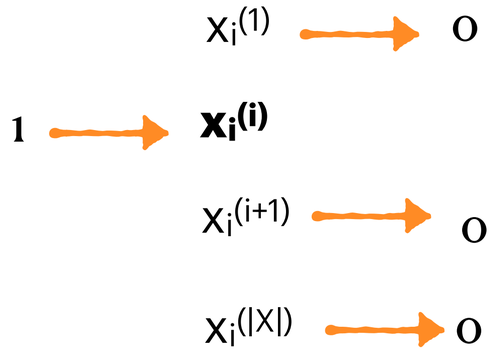
Encode this

Domain {0,1, …,|X|}	
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Forcing the vertices of X in the correct bucket

- Force x_i in bucket i.
- This is done by adding undeletable constraints as shown on right.



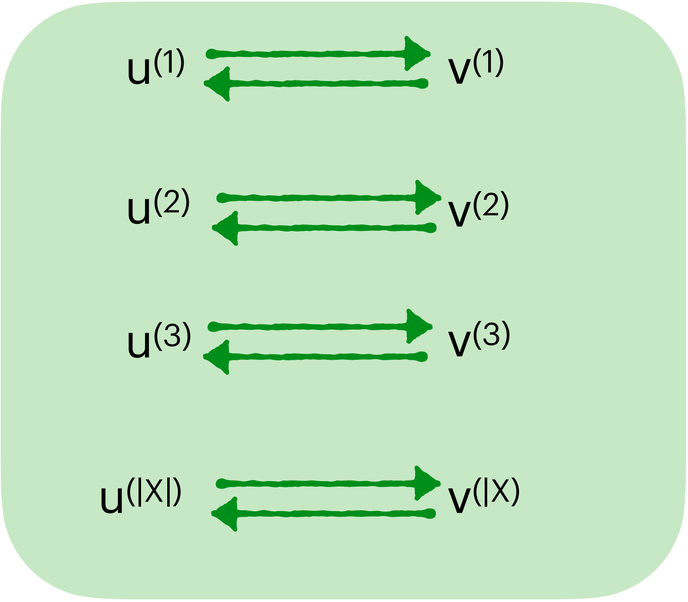
Encode this

Domain {0,1, …,|X|}	
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Forcing an edge uv correctly

- This is done by adding a constraint that is an **AND of all the clauses** shown on right.
- Weight of this constraint=w(uv)



Encode this

Domain {0,1, …,|X|}	
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Forcing the endpoints of a terminal pair st correctly

This is done by adding undeletable constraints shown on right.

$$\neg s^{(i)} \vee \neg t^{(i)}$$
$$\forall 1 \leq i \leq |X|$$

$s^{(1)}$ ..... $t^{(1)}$

$s^{(2)}$ ..... $t^{(2)}$

$s^{(3)}$ ..... $t^{(3)}$

$s^{(|X|)}$ ..... $t^{(|X|)}$

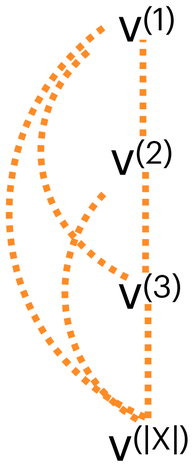
Encode this

Domain {0,1, …,|X|}

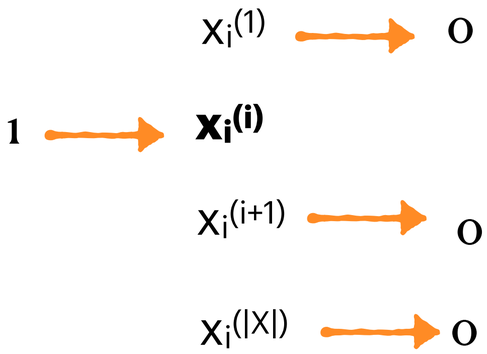
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

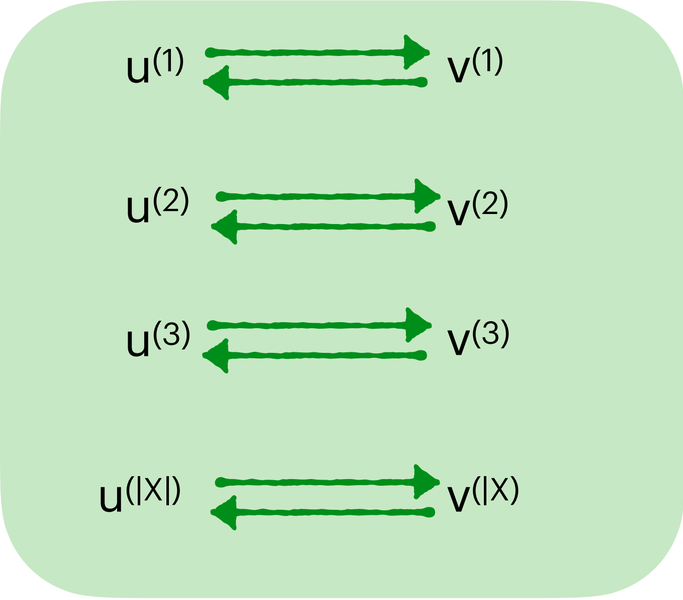
Encoding bucket numbers (domain) in binary



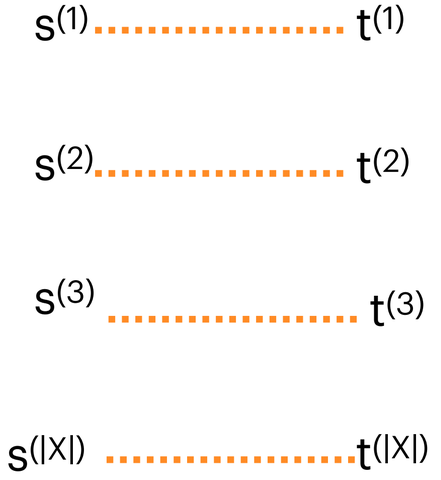
Forcing the vertices of X in the correct bucket



Forcing an edge uv correctly



Forcing the endpoints of a terminal pair st correctly



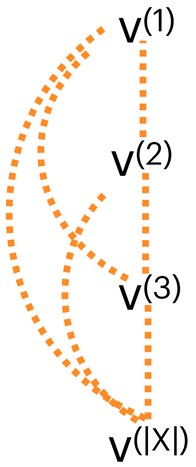
Encode this

Domain {0,1, …,|X|}

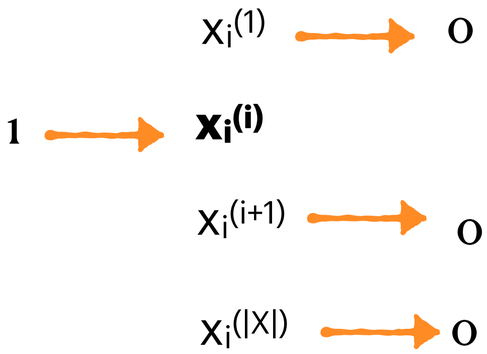
uv ∈ E(G)	**u=v**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

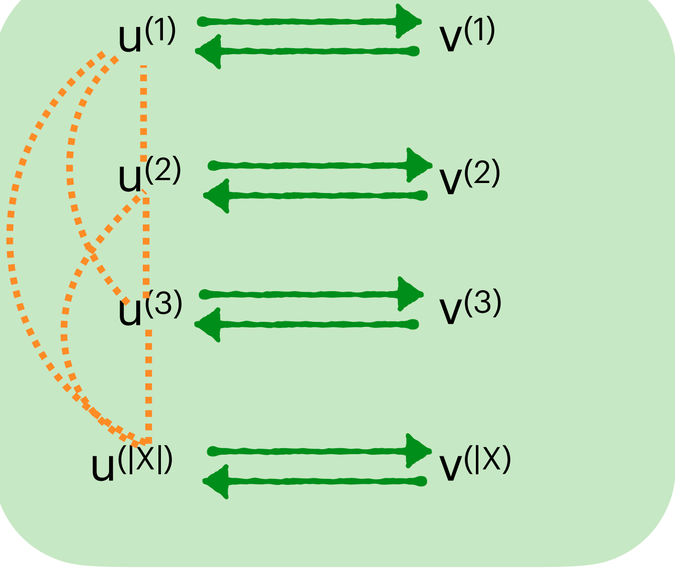
Encoding bucket numbers (domain) in binary



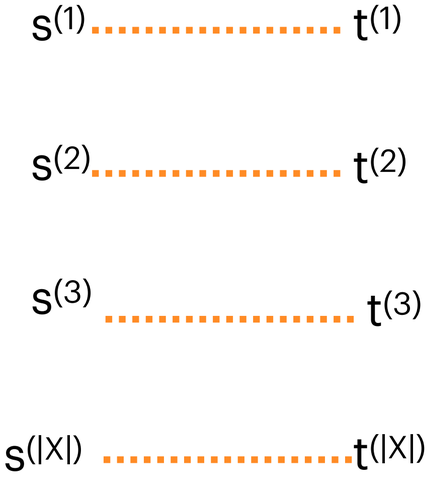
Forcing the vertices of X in the correct bucket



Forcing an edge uv correctly



Forcing the endpoints of a terminal pair st correctly



Undirected Vertex Multicut

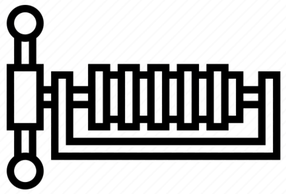
Undirected graph G,
pairs of vertices (terminals) $(s_1, t_1), \ldots, (s_p, t_p)$,
weight function $\mathrm{wt} : V(G) \to \mathbb{N}$,
positive integers k, W

$?\exists$ a set $Z \subseteq V(G) \setminus \{s_i, t_i : i \in [p]\}$
such that $|Z| \leq k$, $\mathrm{wt}(Z) \leq W$ and
for each $i \in \{1, \ldots, p\}$,
$G - Z$ has no $s_i - t_i$ path.

Undirected Vertex Multicut

Undirected graph G,
pairs of vertices (terminals) $(s_1, t_1), \ldots, (s_p, t_p)$,
weight function $\mathrm{wt} : V(G) \to \mathbb{N}$,
positive integers k, W

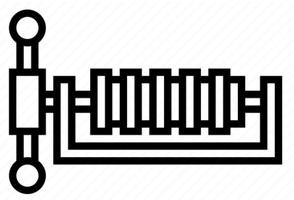
$?\exists$ a set $Z \subseteq V(G) \setminus \{s_i, t_i : i \in [p]\}$
such that $|Z| \leq k$, $\mathrm{wt}(Z) \leq W$ and
for each $i \in \{1, \ldots, p\}$,
$G - Z$ has no $s_i - t_i$ path.



Iterative compression

Input++: $X \subseteq V(G)$, $|X| \leq k+1$, every terminal pair path intersects X.

Assume WLOG: $X = \{x_1, \ldots, x_{|X|}\}$, X is an independent set and $Z \cap X = \emptyset$.



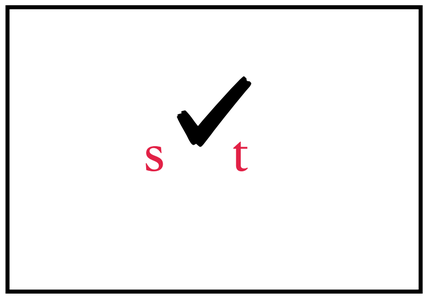
Iterative compression

Weighted Vertex Multicut

With Iterative Compression

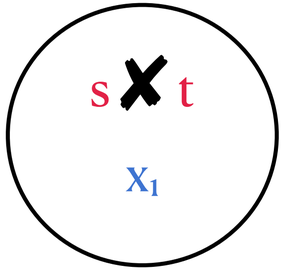
Z

G-Z



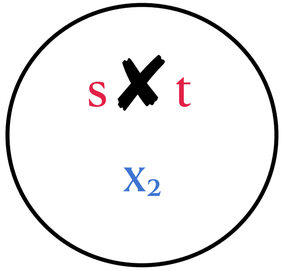
0

connected components that do not contain X



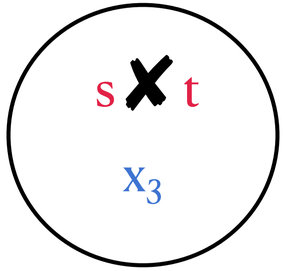
1

contains x_1



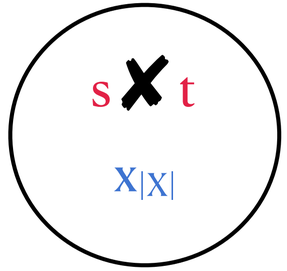
2

contains x_2



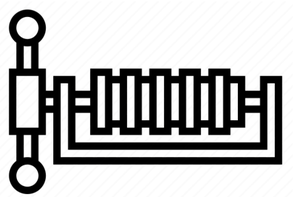
3

contains x_3



|X|

contains $x_{|X|}$



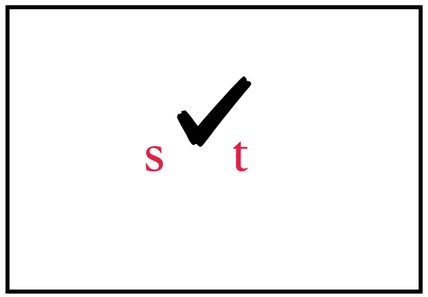
Iterative compression

Weighted Vertex Multicut

With Iterative Compression

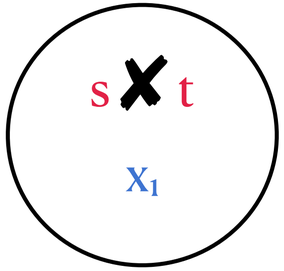
Z

G-Z



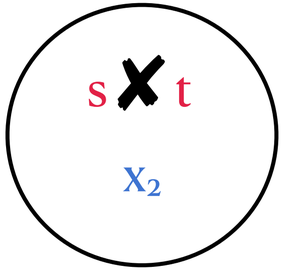
0

connected components that do not contain X



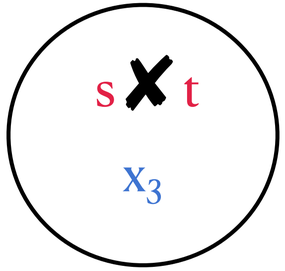
1

contains x_1



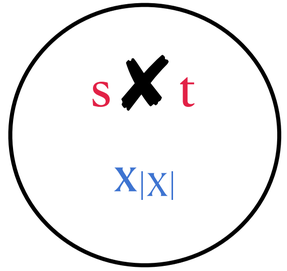
2

contains x_2



3

contains x_3



|X|

contains $x_{|X|}$

Domain {0,1, …,|X|}

uv ∈ E(G)	**“u=v”**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

Encode this

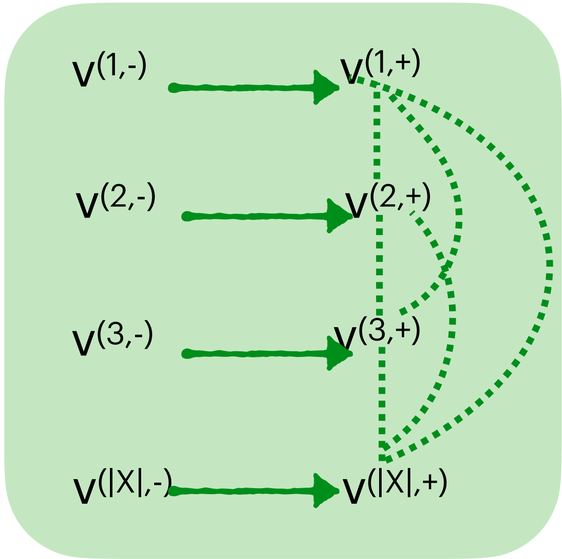
Domain {0,1, …,|X|}	
uv ∈ E(G)	**"u=v"**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Encoding bucket numbers of vertices (domain) in Boolean

- Create **|X|+|X| variables** for every vertex v.
- Add the **AND of all the clauses** on the right as a constraint.
- A pair ($v^{(i,-)}$, $v^{(l,+)}$) is selected if it has assignment (0,1) or (1,1).
- Weight of this constraint=wt(v)

At most one pair is selected and if none is selected then all pairs get assignment (0,0).



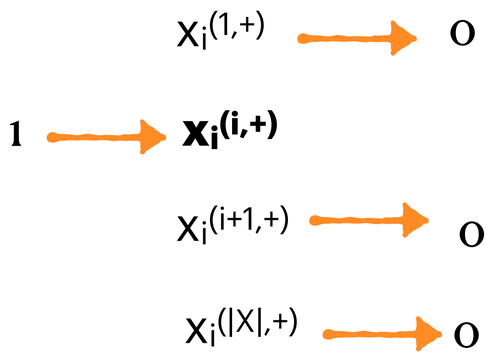
Encode this

Domain {0,1, …,|X|}	
uv ∈ E(G)	**“u=v”**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Forcing the vertices of X in the correct bucket

- Force x_i in bucket i.
- This is done by adding undeletable constraints as shown on right.



Encode this

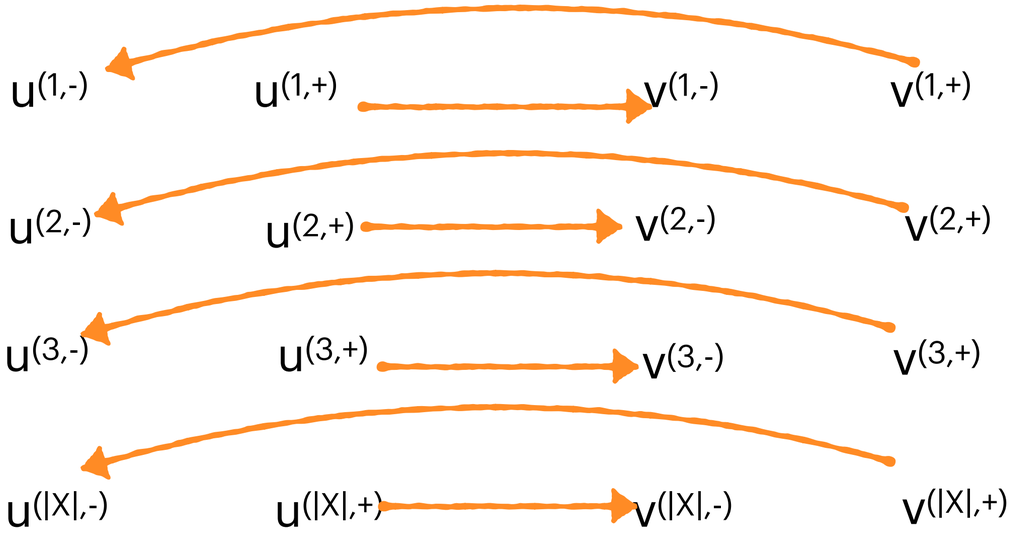
Domain {0,1, …,|X|}	
uv ∈ E(G)	**"u=v"**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Forcing an edge uv correctly

This is done by adding undeletable constraints shown on right.

- If none of the constraints for the variables u or is deleted, then observe that the assignment of u and v to buckets corresponding to selected pair, is valid.
- Say the constraint for variable u was deleted. Then set $u^{(i,-)}$=1 and $u^{(i,+)}$=0 for all i.



Encode this

Domain {0,1, …,|X|}	
uv ∈ E(G)	**“u=v”**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

Forcing the endpoints of a terminal pair st correctly

This is done by adding undeletable constraints shown on right.

$s^{(1,+)}$ ..... $t^{(1,+)}$

$s^{(2,+)}$ ..... $t^{(2,+)}$

$s^{(3,+)}$ ..... $t^{(3,+)}$

$s^{(|X|,+)}$ ..... $t^{(|X|,+)}$

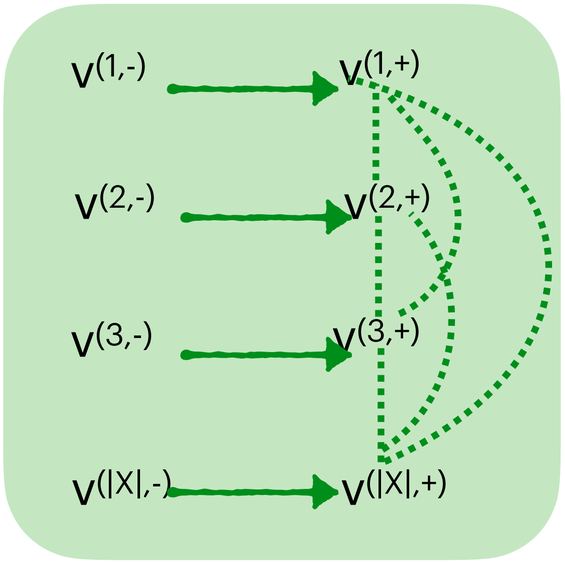
Encode this

Domain {0,1, …,|X|}

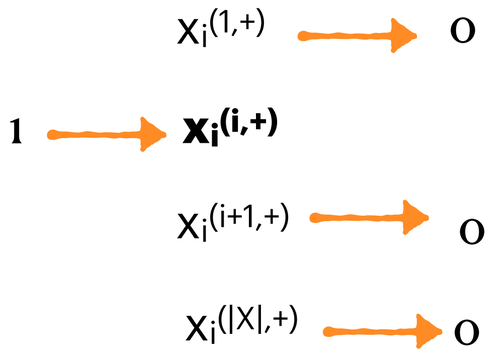
uv ∈ E(G)	**“u=v”**
st ∈ T	**s ≠ t or s=t=0**
$x_i \in X$	**x_i=i**

as MinCSP(Γ_{good})

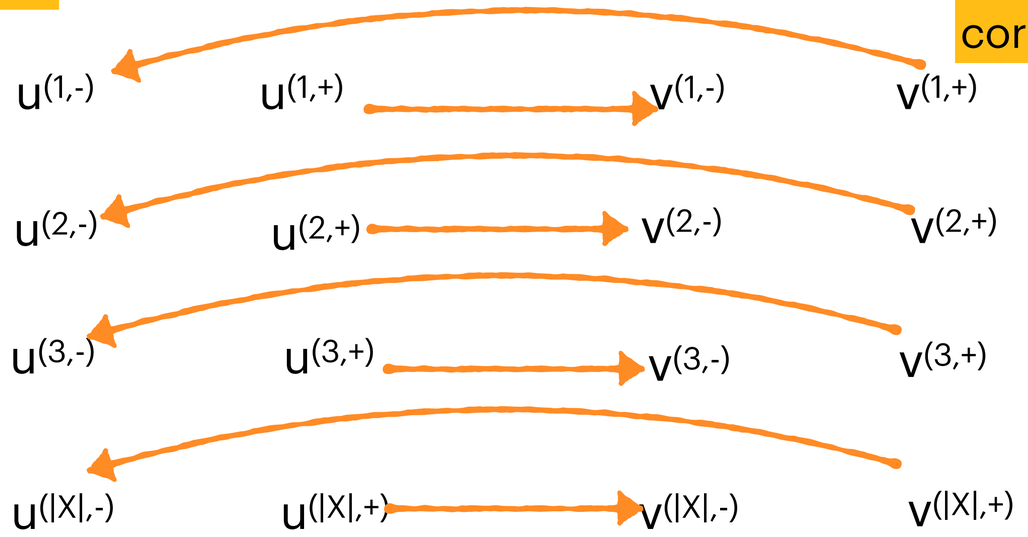
Encoding bucket numbers (domain) in Boolean



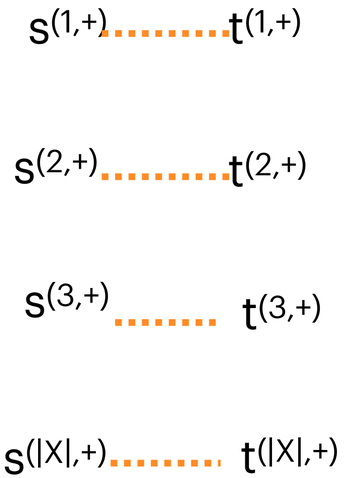
Forcing the vertices of X in the correct bucket



Forcing an edge uv correctly



Forcing the endpoints of a terminal pair st correctly



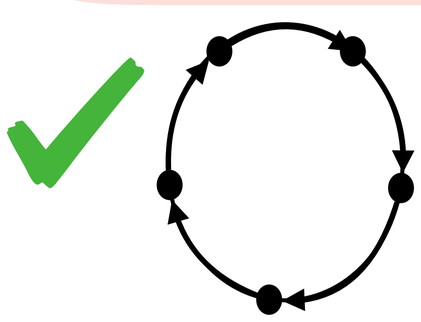
Weighted Subset DFAS

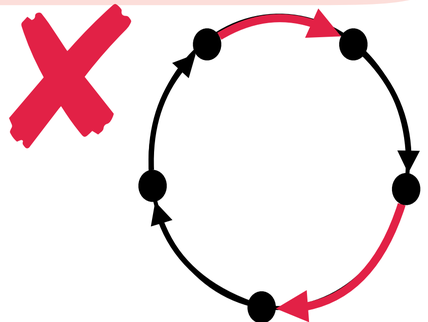
Directed graph G,
red arcs $R \subseteq E(G)$,
wt : $E(G) \to \mathbb{N}$
positive integers
k, W

$?\exists$ a set $Z \subseteq E(G)$
such that $|Z| \leq k$, $\text{wt}(Z) \leq W$, and
$G - Z$ has no directed cycle with at least one red arc.

$\equiv$

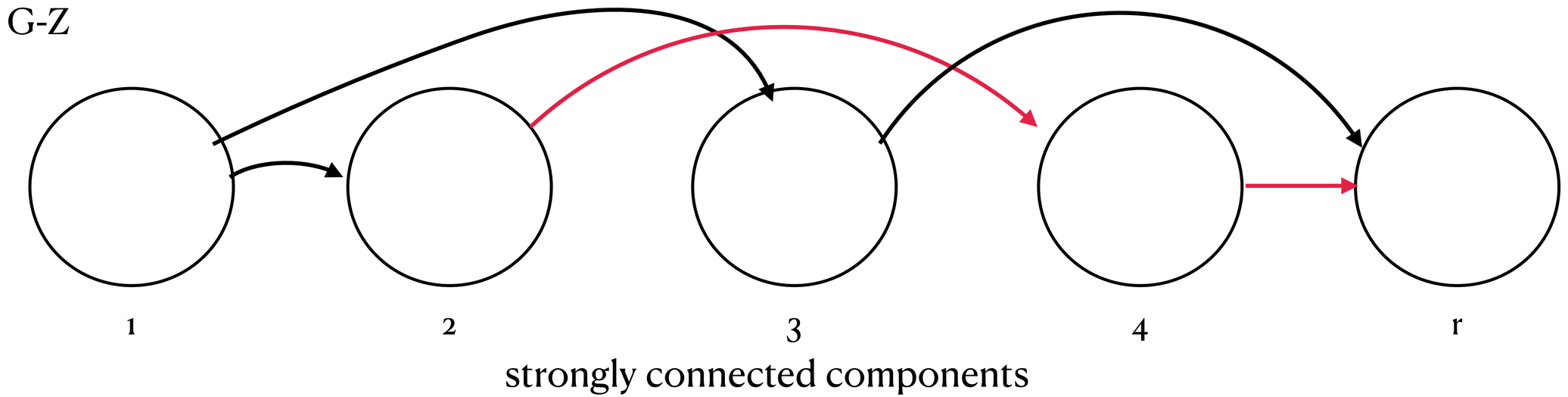
for each $st \in R$, s and t are in different strongly connected components of $G - Z$.





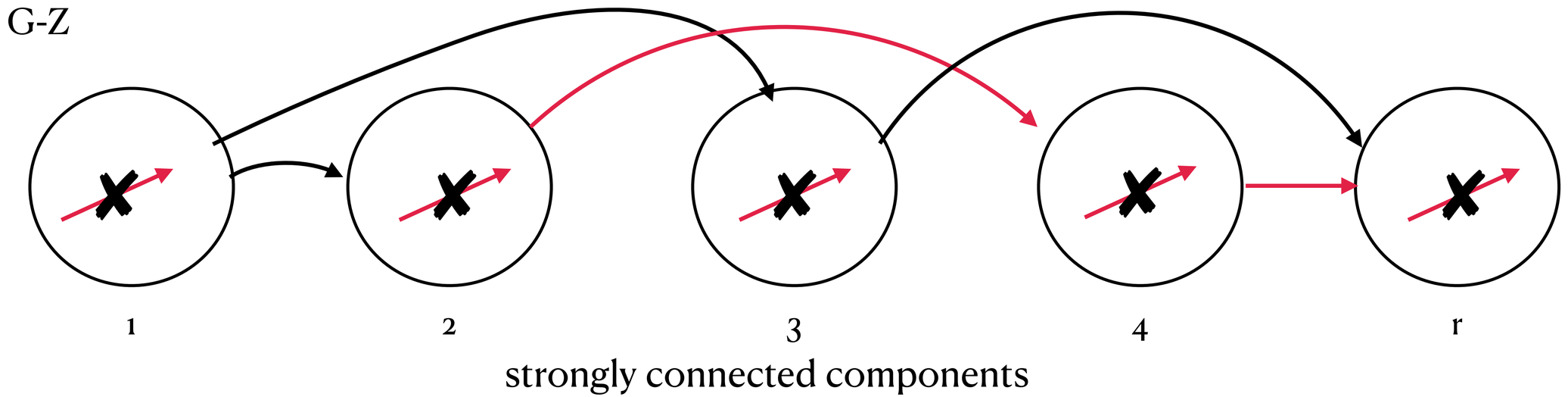
WEIGHTED SUBSET DFAS

WEIGHTED MINCSP($\leq$, $<$)



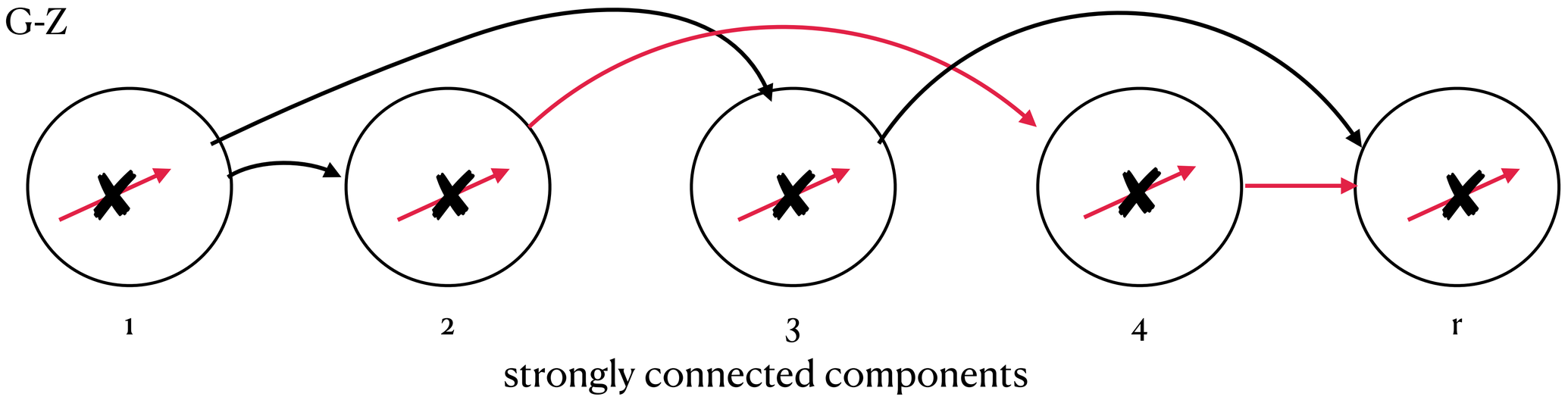
Weighted Subset DFAS

Weighted MinCSP($\leq$, $<$)

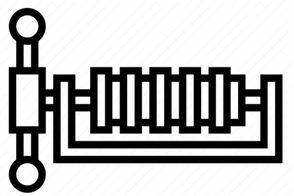


Weighted Subset DFAS

Weighted MinCSP($\leq$, $<$)



For every vertex $v \in V(G)$,	create a variable v	domain is 1,....,r
For every edge $uv \in E(G) \setminus R$,	create a constraint $\mathbf{u \leq v}$	weight=w(uv)
For every red arc $st \in R$,	create an constraint $\mathbf{s < t}$	weight=w(st)

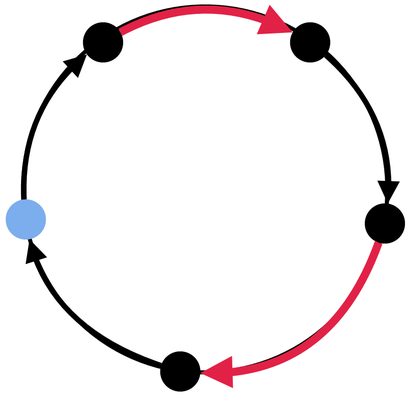


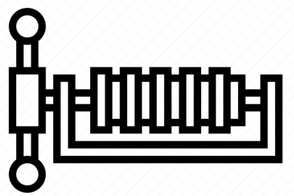
Weighted Subset DFAS

with Iterative Compression

n

Input++: $X \subseteq V(G)$, $|X| \leq k+1$, every cycle containing a red arc intersects X.

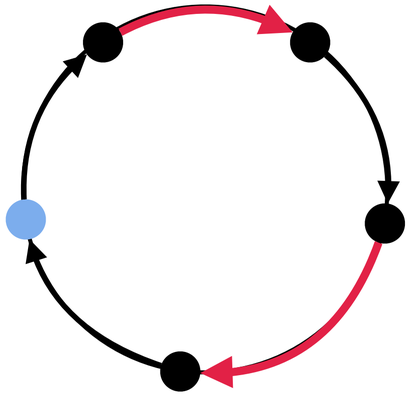




Weighted Subset DFAS

with Iterative Compression

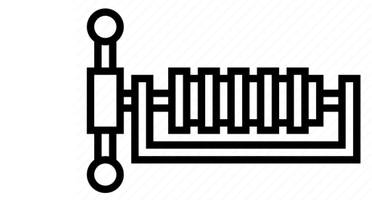
Input++: $X \subseteq V(G)$, $|X| \leq k+1$, every cycle containing a red arc intersects X.



Assume WLOG: $X=\{x_1,\ldots,x_{|X|}\}$ and there is no path from x_j to x_i, $j > i$, in G-Z.

n

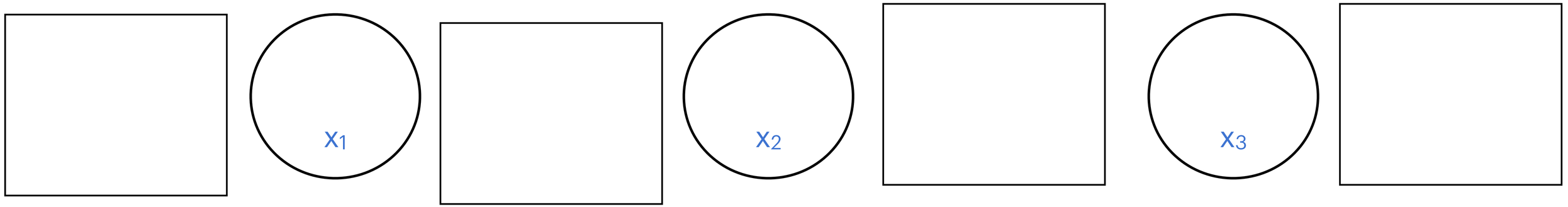
$2^{\mathcal{O}(k \log k)}$



Weighted Subset DFAS

with Iterative Compression

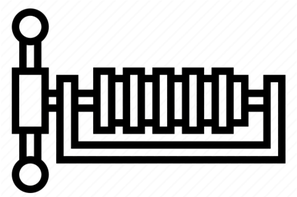
G-Z



1 2 3 4 5 6 2|X|+1

Set of strongly connected components between x_1 and x_2

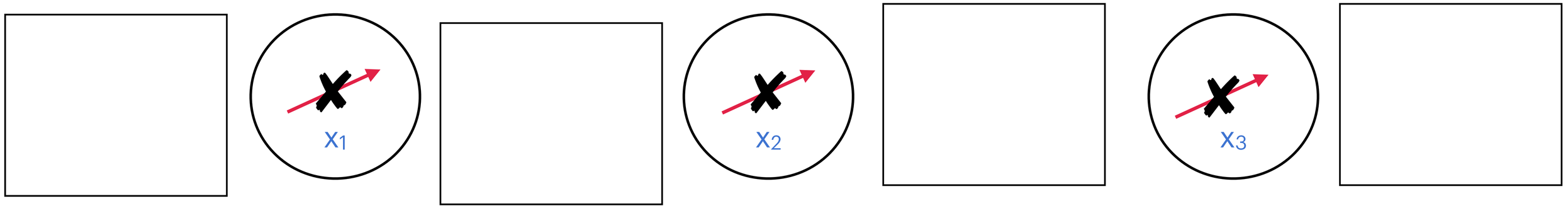
Strongly connected component containing x_2



Weighted Subset DFAS

with Iterative Compression

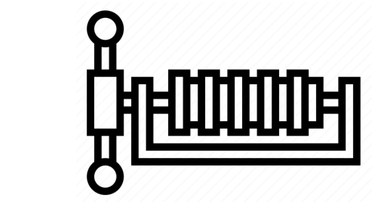
G-Z



1 2 3 4 5 6 2|X|+1

Set of strongly connected components between x_1 and x_2

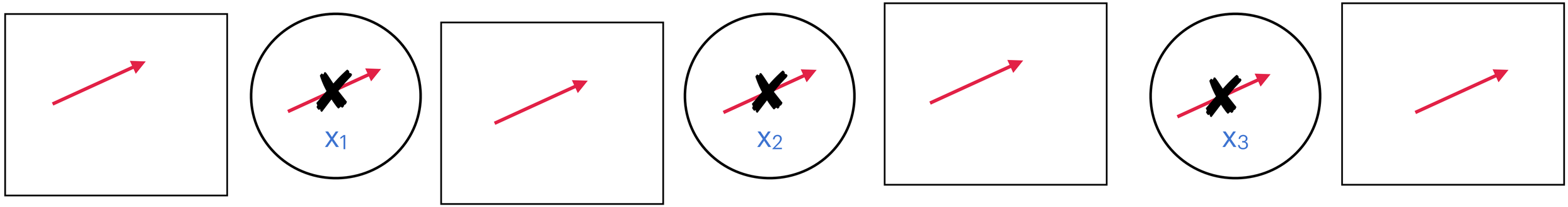
Strongly connected component containing x_2



Weighted Subset DFAS

with Iterative Compression

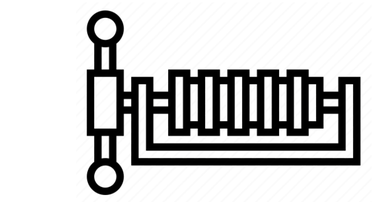
G-Z



1 2 3 4 5 6 2|X|+1

Set of strongly connected components between x_1 and x_2

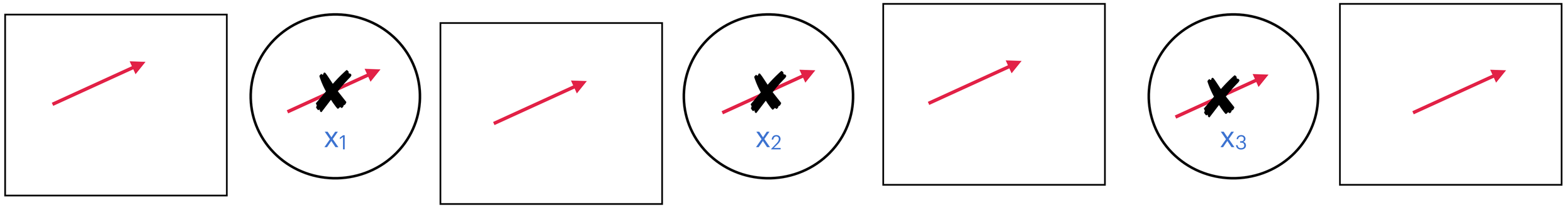
Strongly connected component containing x_2



Weighted Subset DFAS

with Iterative Compression

G-Z



1 2 3 4 5 6 2|X|+1

Set of strongly connected components between x_1 and x_2

Strongly connected component containing x_2

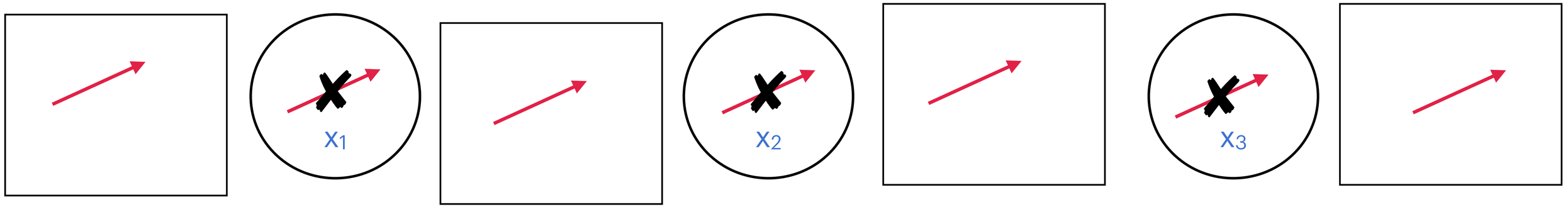
A **non-red arc** can be inside a circle or inside a rectangle, or as a forward arc.

A red arc **can be inside a rectangle**, or as a forward arc (but not inside a circle).

Weighted Subset DFAS

with Iterative Compression

G-Z



1 2 3 4 5 6 2|X|+1

Set of strongly connected components between x_1 and x_2

Strongly connected component containing x_2

A **non-red arc** can be inside a circle or inside a rectangle, or as a forward arc.

A red arc **can be inside a rectangle**, or as a forward arc (but not inside a circle).

Domain {1, …,2|X|+1}

$uv \in E(G)\setminus R$	**$u \le v$**
$st \in R$	**$s < t$ or $s=t=i$ where i is odd**
$x_i \in X$	**$x_i=2i$**

Encode this

Domain {1, …,2|X|+1}	
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

Encoding bucket numbers (domain) in binary

Encode this

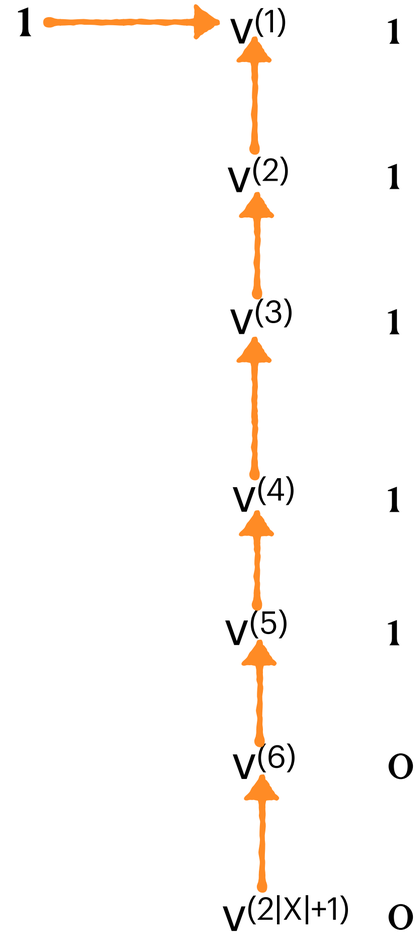
Domain {1, …,2|X|+1}	
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

Encoding bucket numbers (domain) in binary

- Create **2|X|+1 variables** for every vertex v.
- Ensure that if v belongs to bucket, say 5, in G-Z, then the valid assignment sets the variables ($v^{(1)}$,$v^{(2)}$,$v^{(3)}$,$v^{(4)}$,$v^{(5)}$,$v^{(6)}$,$v^{(2|X|+1)}$) to (1,1,1,1,1,0,0).
- This can be ensured by adding undeletable constraints as shown on right.

The highest superscript that gets assigned 1, tells the bucket this vertex will go into.



Encode this

Domain {1, …,2|X|+1}	
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

Forcing the vertices of X in the correct bucket

- Force x_i in bucket 2i.
- This is done by adding undeletable constraints as shown on right.

1 → $x_i^{(1)}$

1 → $x_i^{(2)}$

1 → $x_i^{(2i-1)}$

1 → **$x_i^{(2i)}$**

$x_i^{(2i+1)}$ → 0

$x_i^{(2|X|)}$ → 0

$x_i^{(2|X|+1)}$ → 0

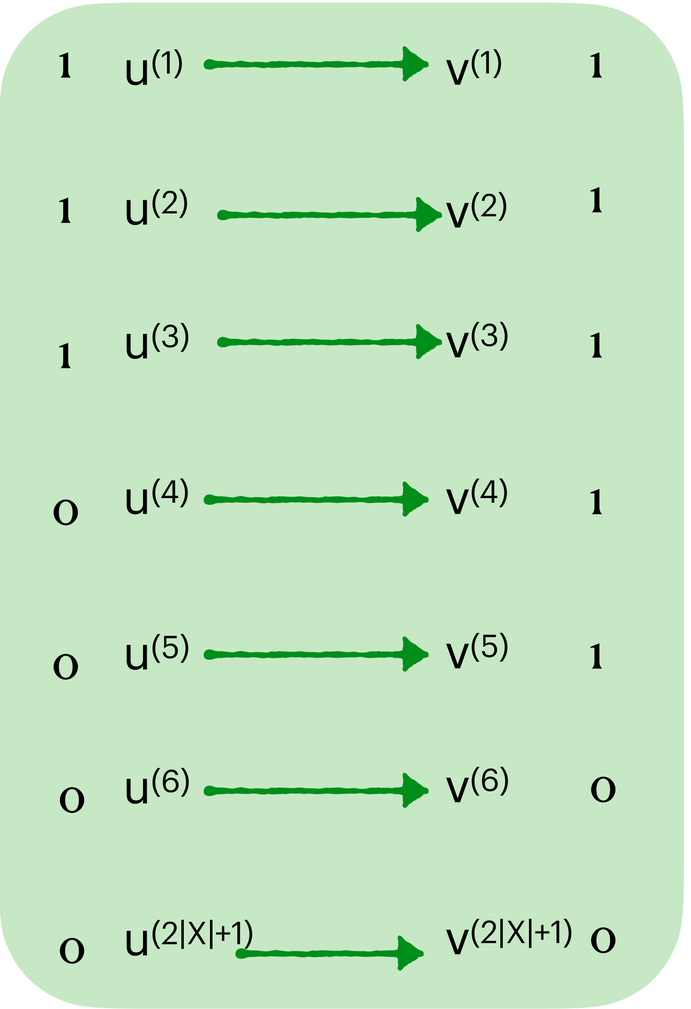
Encode this

Domain {1, …,2|X|+1}	
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

Forcing a non-red arc uv correctly

- This is done by adding a constraint that is an **AND of all the clauses** shown on right.
- Weight of this constraint=w(uv)



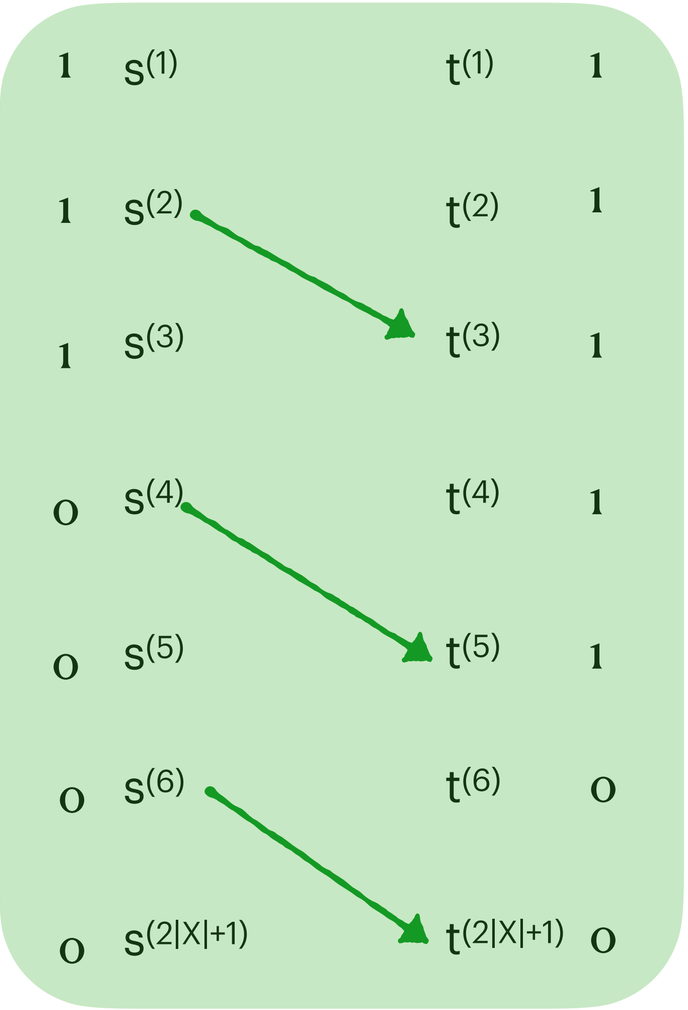
Encode this

Domain {1, …,2|X|+1}	
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

Forcing a red arc st correctly

- This is done by adding a constraint that is an **AND of all the clauses** shown on right.
- Weight of this constraint=w(st)



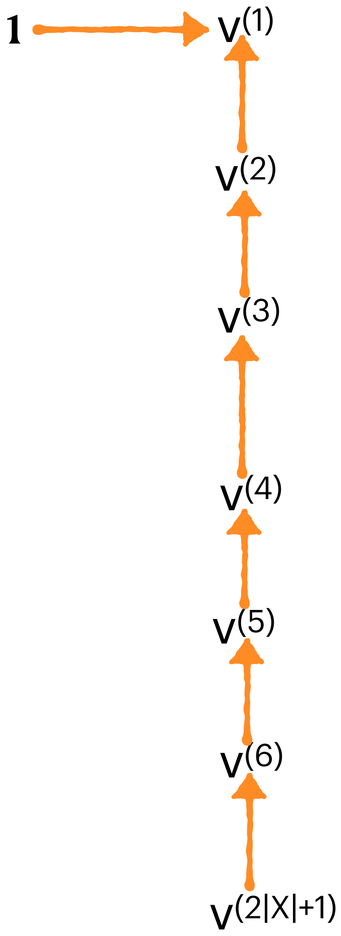
Encode this

Domain {1, …,2|X|+1}

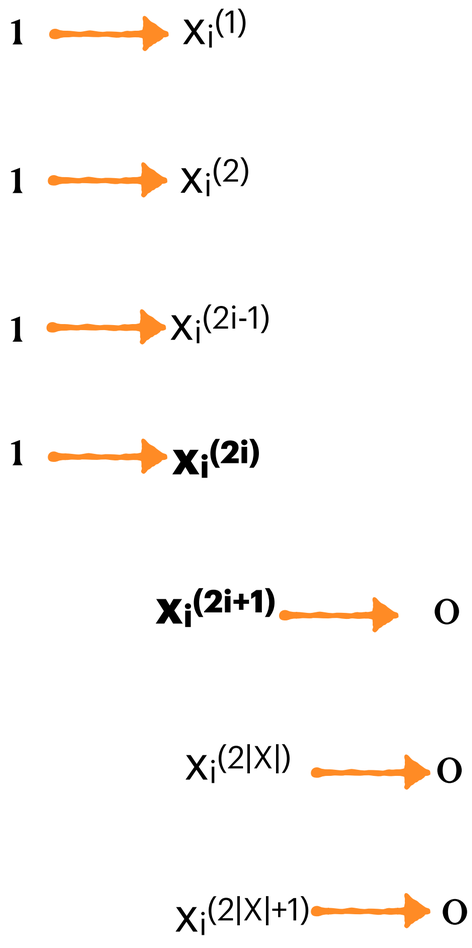
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

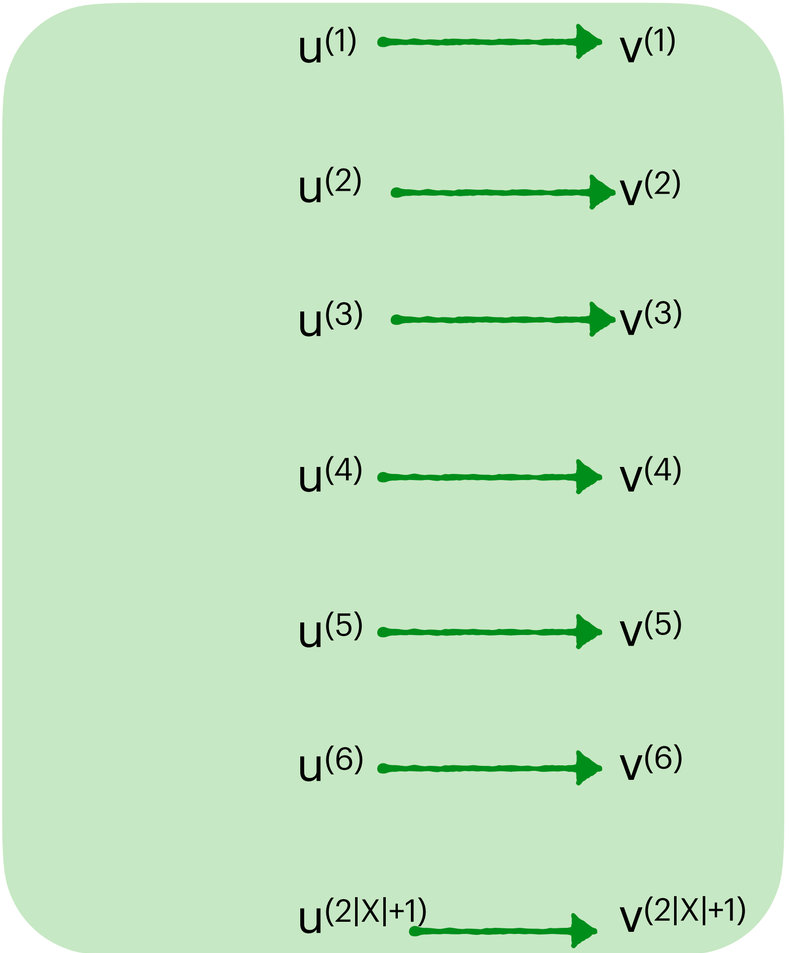
Encoding bucket numbers (domain) in binary



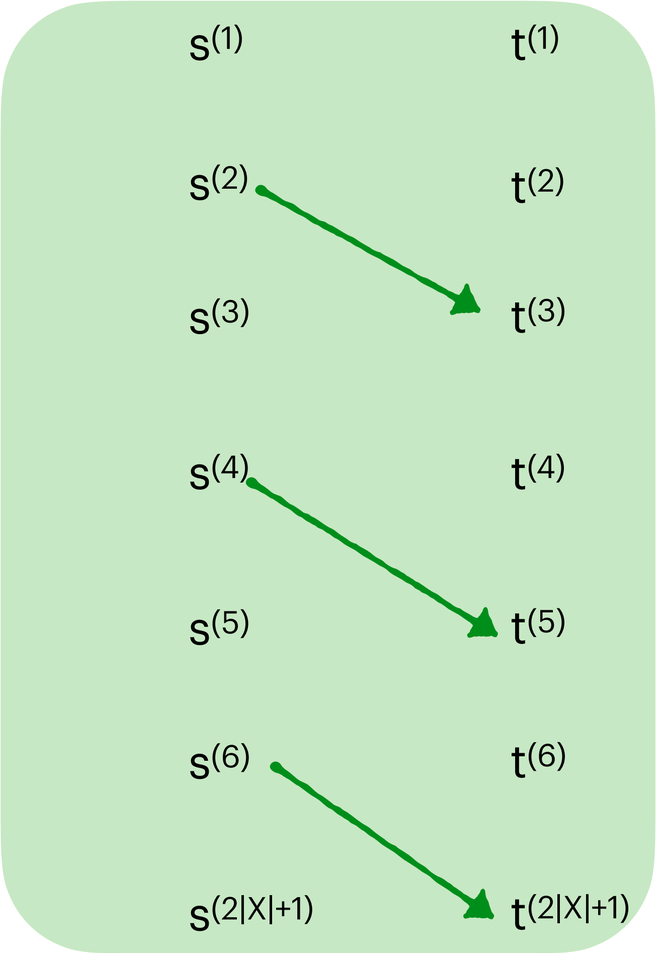
Forcing the vertices of X in the correct bucket



Forcing a non-red edge uv correctly



Forcing a red edge st correctly



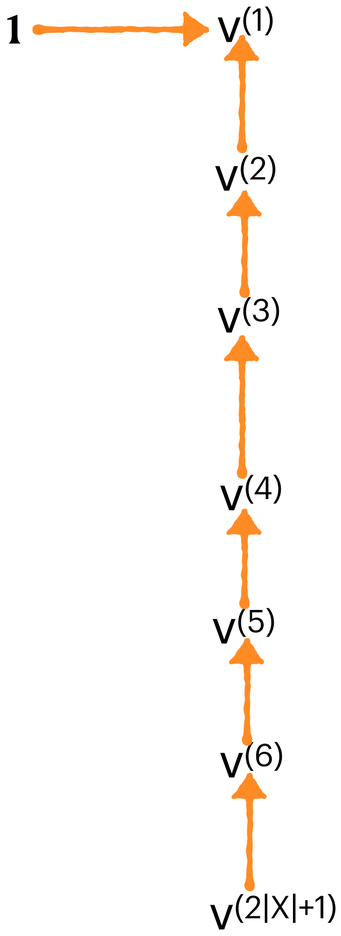
Encode this

Domain {1, …,2|X|+1}

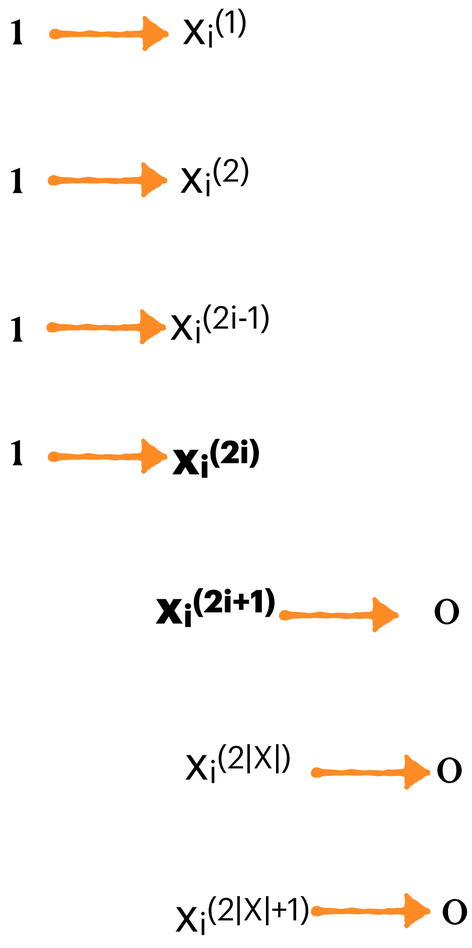
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

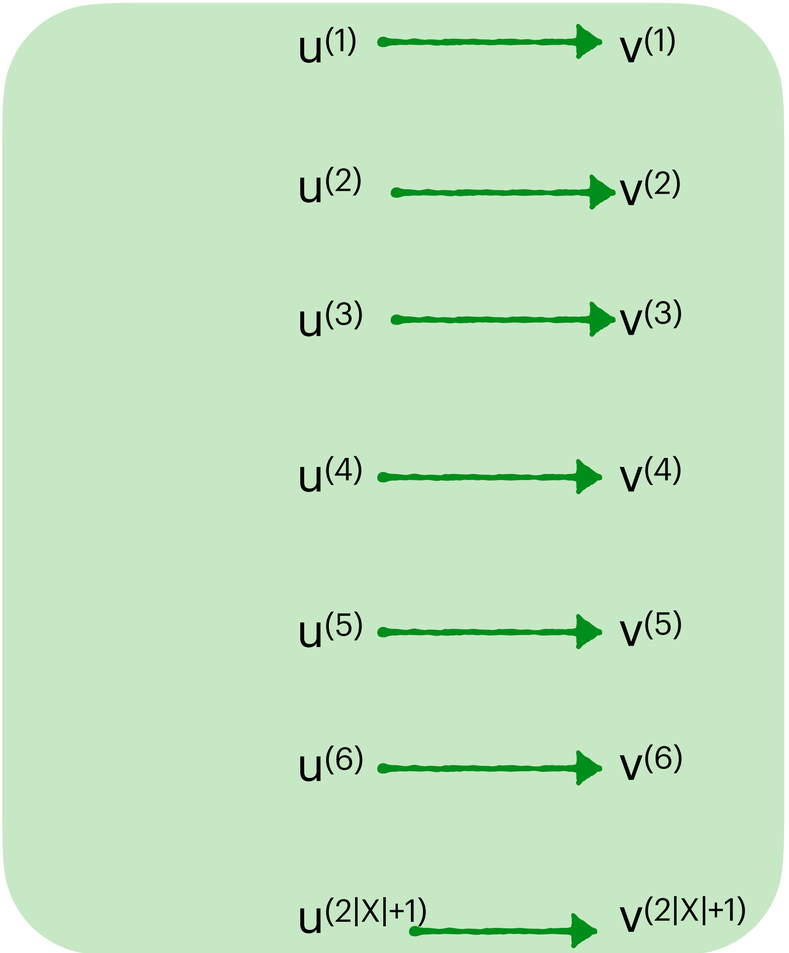
Encoding bucket numbers (domain) in binary



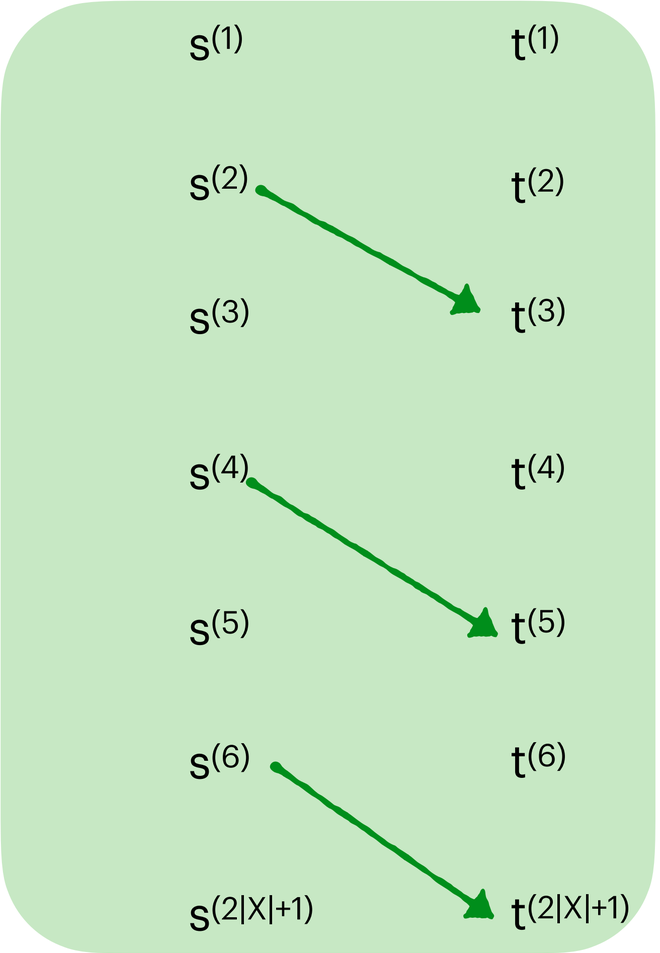
Forcing the vertices of X in the correct bucket



Forcing a non-red edge uv correctly



Forcing a red edge st correctly



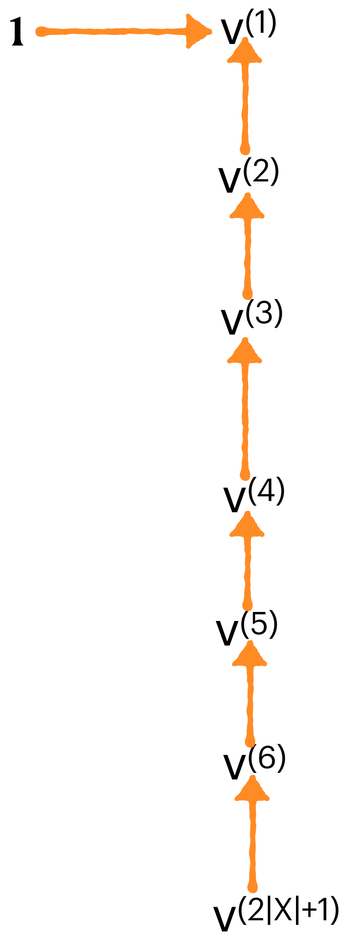
Encode this

Domain {1, …,2|X|+1}

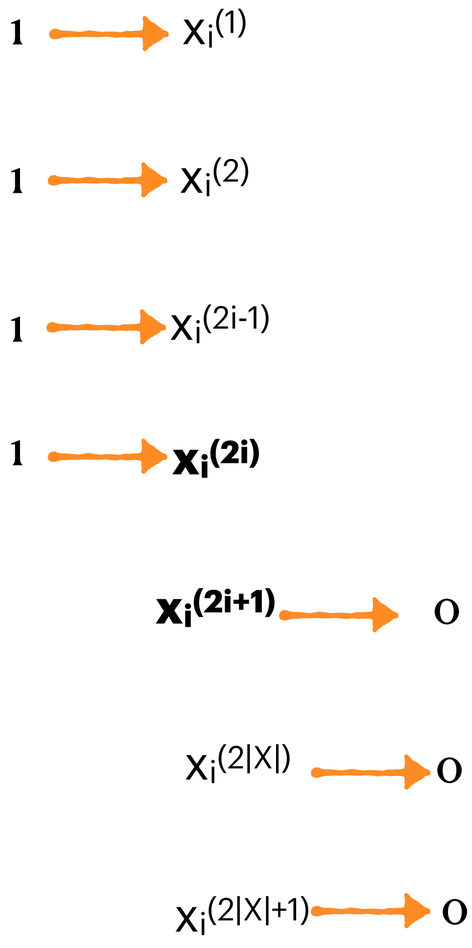
uv ∈ E(G)\R	**u≤v**
st ∈ R	**s < t or s=t=i where i is odd**
$x_i \in X$	**x_i=2i**

as MinSAT(Γ_{good})

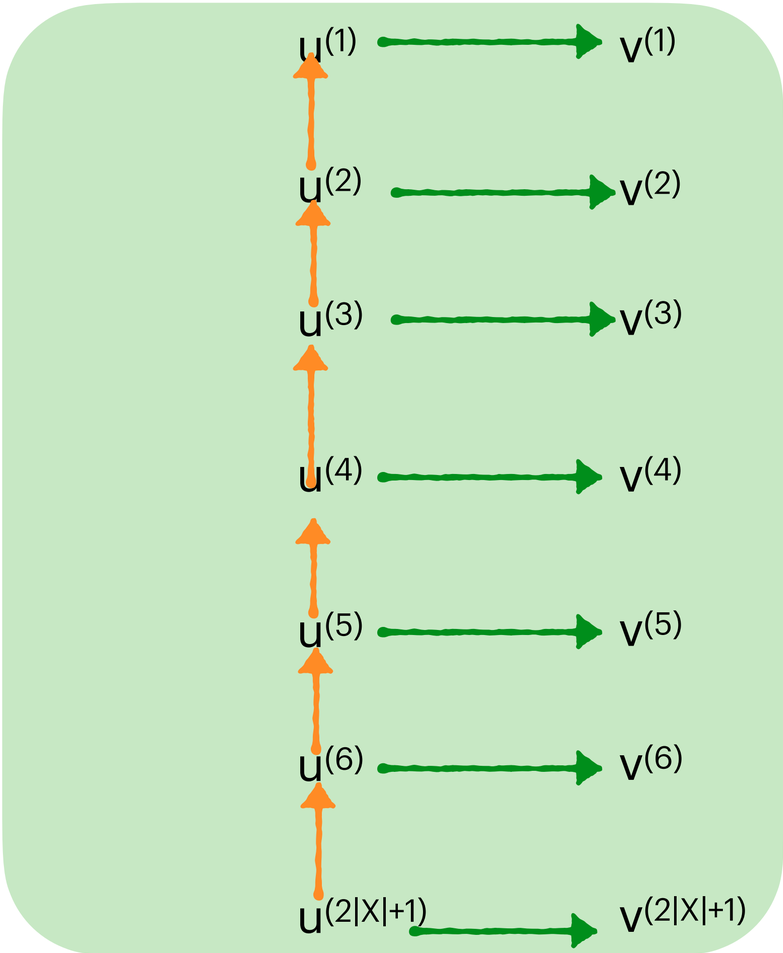
Encoding bucket numbers (domain) in binary



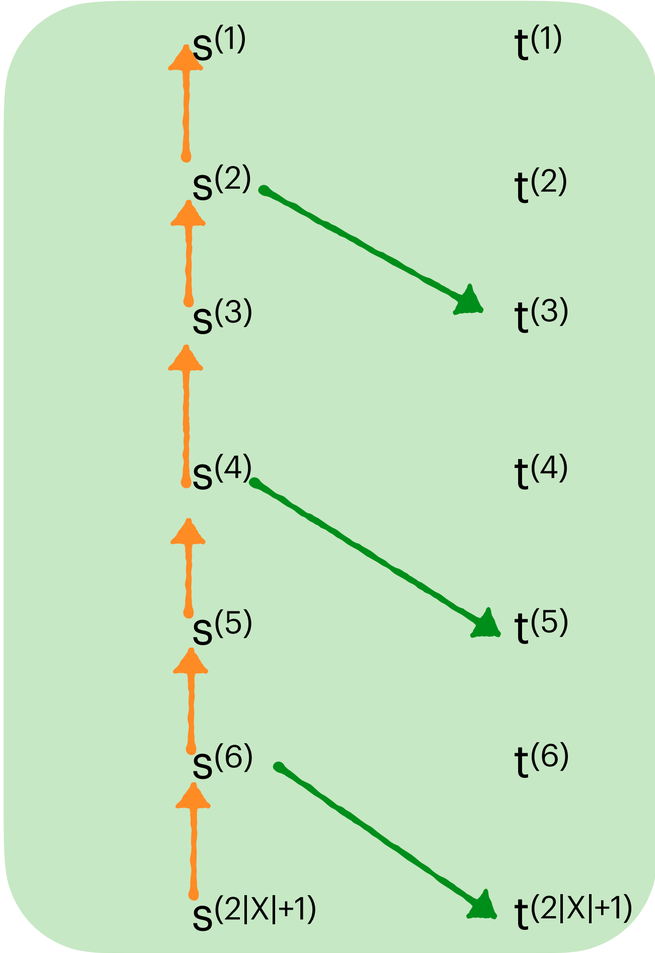
Forcing the vertices of X in the correct bucket



Forcing a non-red arc uv correctly



Forcing a red arc st correctly



Beyond Boolean MinCSP

Min CSP
over Point Algebra

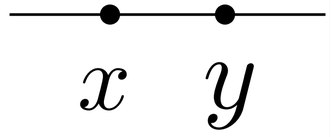
Point Algebra($=$, $\neq$, $<$, $\leq$)

Domain $\mathbb{Q}$

Constraints have access to

$<$, $=$, $\leq$, $\neq$

and are FO formulae

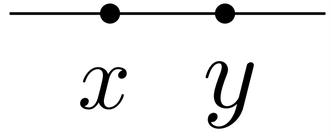


Min CSP
over Point Algebra

≡

Point Algebra($=$, $\neq$, $<$, $\leq$)
Domain $\mathbb{Q}$

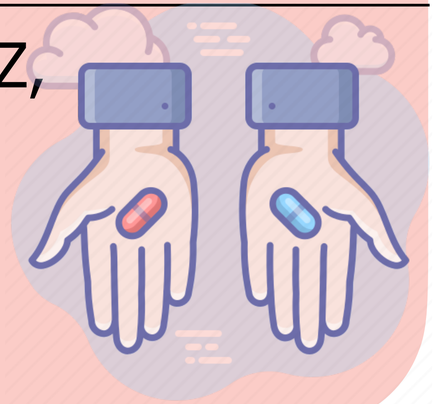
Constraints have access to
$<$, $=$, $\leq$, $\neq$
and are FO formulae



Directed Symmetric Multicut

Given a digraph G,
pairs $(s_1, t_1), \ldots, (s_p, t_p)$
integer k

Delete at most k arcs, say Z,
such that in $G - Z$,
for every $i \in \{1, \ldots, p\}$,
either no $s_i \to t_i$ path, or no $t_i \to s_i$ path.

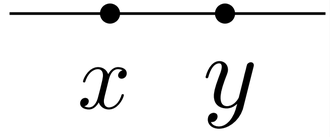


Min CSP
over Point Algebra

≡

Point Algebra($=$, $\neq$, $<$, $\leq$)
Domain $\mathbb{Q}$

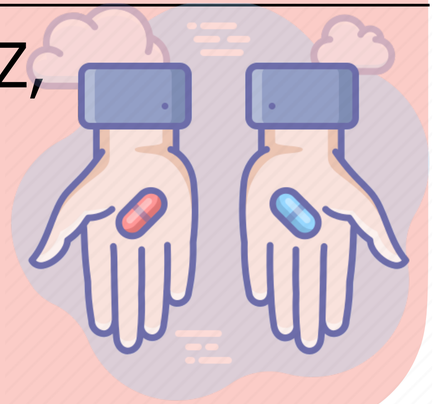
Constraints have access to
$<$, $=$, $\leq$, $\neq$
and are FO formulae



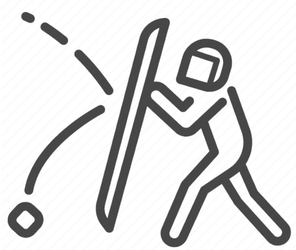
Directed Symmetric Multicut

Given a digraph G,
pairs $(s_1, t_1), \ldots, (s_p, t_p)$
integer k

Delete at most k arcs, say Z,
such that in $G - Z$,
for every $i \in \{1,\ldots,p\}$,
either no $s_i \to t_i$ path, or no $t_i \to s_i$ path.



Is this FPT
parameterized by k?



MIN CSP
over basic Allen Algebra

Domain: intervals $\{[a, b] : a, b \in \mathbb{Q}, a < b\}$

Basic constraints: “before”, “equals”, “meets”, “overlaps”, “contains”, “starts”, “finishes”.

[Dabrowski, Jonsson, Ordyniak, Osipov, Pilipczuk, **Sharma,** IPEC 2023]
FPT v/s W[1]-hard dichotomy based on which subset of the above 7 constraints are allowed.
In general, admits 2-approximation in FPT time.

Summary

Before flow-augmentation

With flow-augmentation

Beyond flow-augmentation?

Summary

Before flow-augmentation	With flow-augmentation	Beyond flow-augmentation?

Greedy tools (directed)

Important cuts,

Shadow Removal

Summary

Before flow-augmentation	With flow-augmentation	Beyond flow-augmentation?

Greedy tools (directed)

Important cuts,

Shadow Removal

DIRECTED MULTICUT

Boolean MinCSP dichotomy

Summary

Before flow-augmentation

Greedy tools (directed)

Important cuts,

Shadow Removal

With flow-augmentation

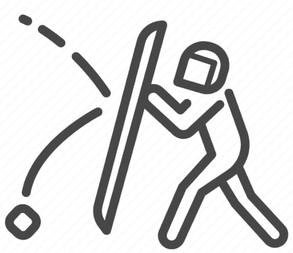
DIRECTED MULTICUT

Boolean MinCSP dichotomy

Beyond flow-augmentation?

MIN CSP
over Point Algebra

Summary



Before flow-augmentation

Greedy tools (directed)

Important cuts,

Shadow Removal

With flow-augmentation

DIRECTED MULTICUT

Boolean MinCSP dichotomy

Beyond flow-augmentation?

MIN CSP
over Point Algebra