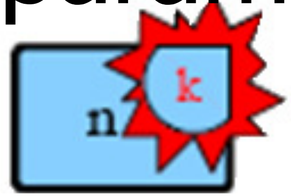
Flow-augmentation:

advances in parameterized (weighted) cut problems



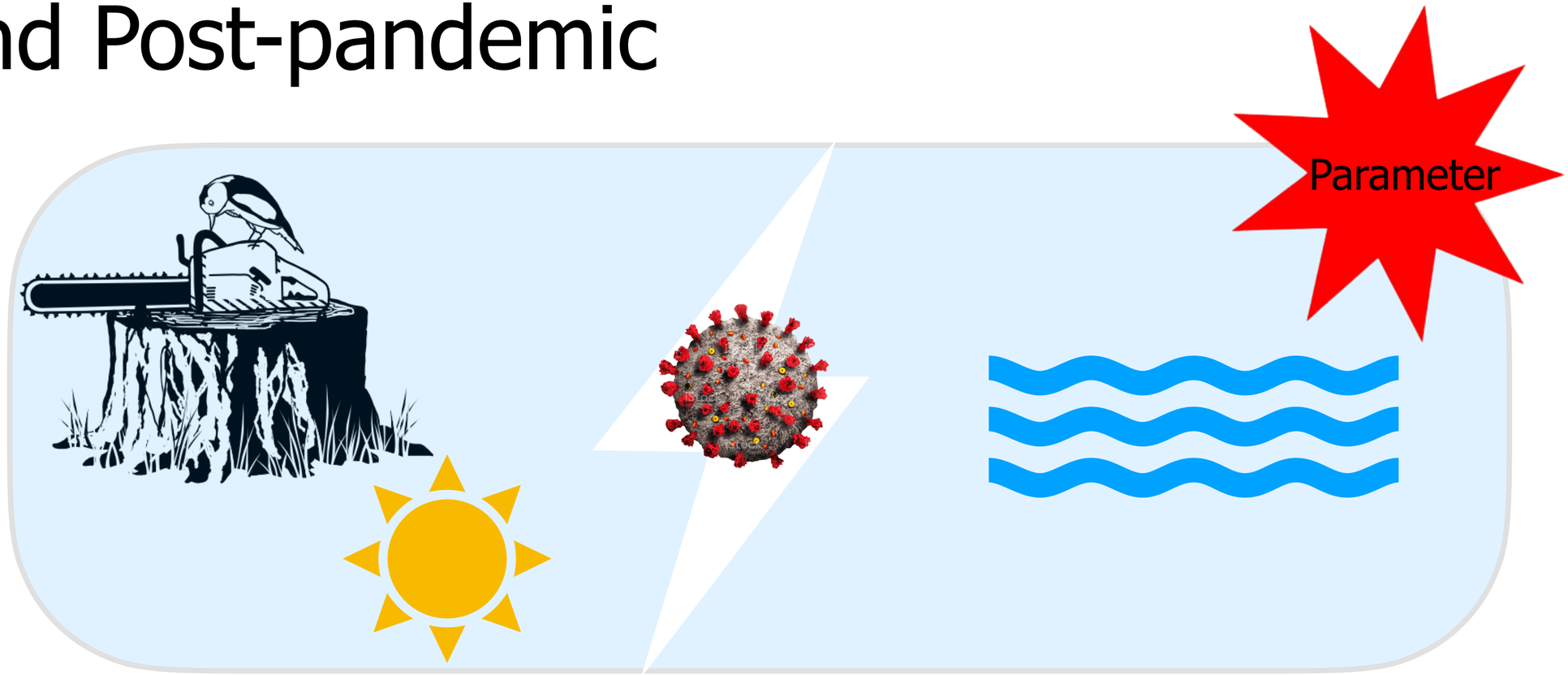
Recent Trends in Algorithms
NISER Bhubaneswar

Roohani Sharma

July 27, 2023

max planck institut informatik

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



Cuts, flows and CSPs

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 (MC)

(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 (MC)

(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) MultiwayCut (MWC)

(Directed) graph G,
pairs of 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

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.

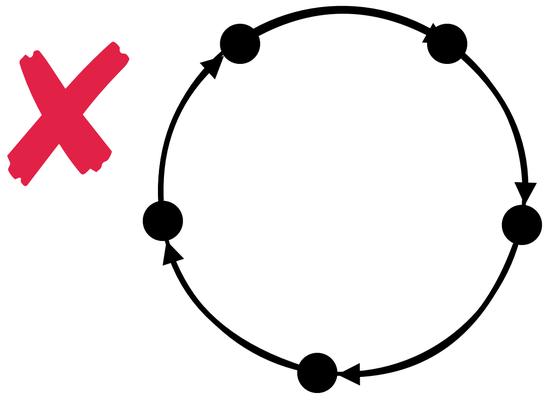
NP-hard

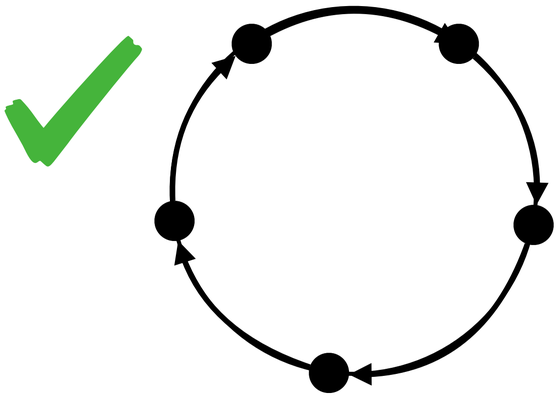
Subset DFAS

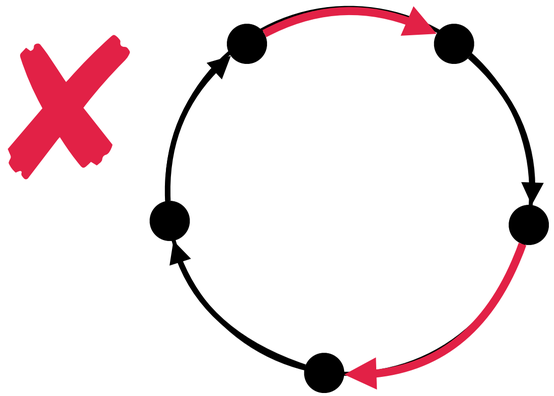
Directed graph G,
Red arcs $R \subseteq E(G)$,
positive integer k

$?\exists$ 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

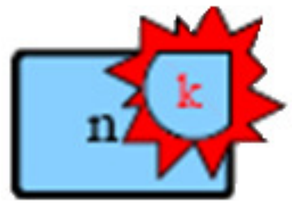






Parameterized Complexity

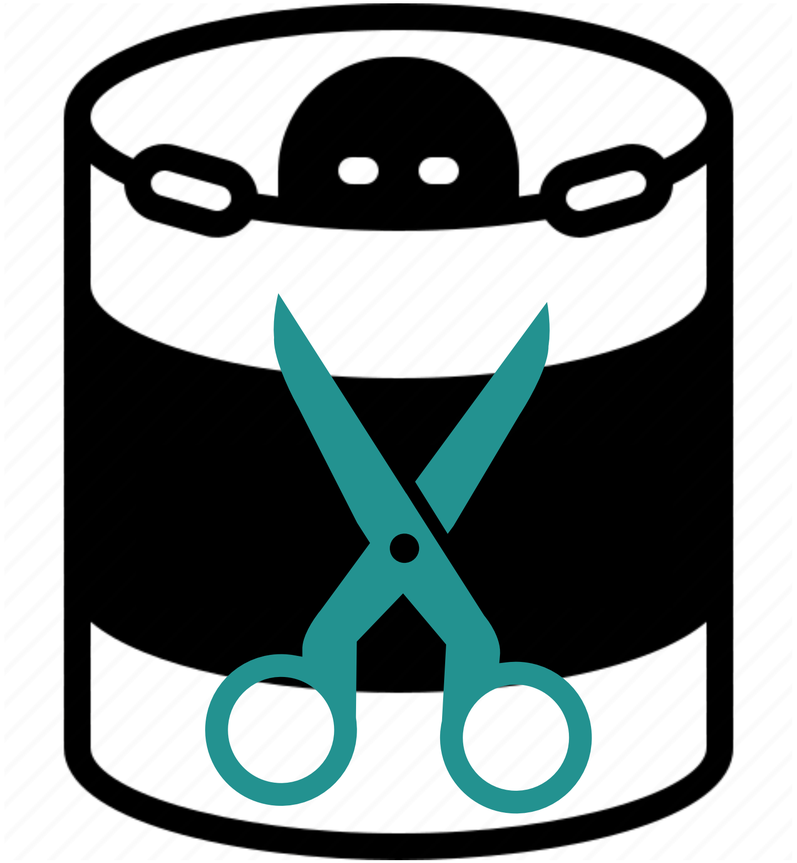
— Parameter is solution size k



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

W[1]-hard, otherwise.

Cut problems remained elusive for a very long time!



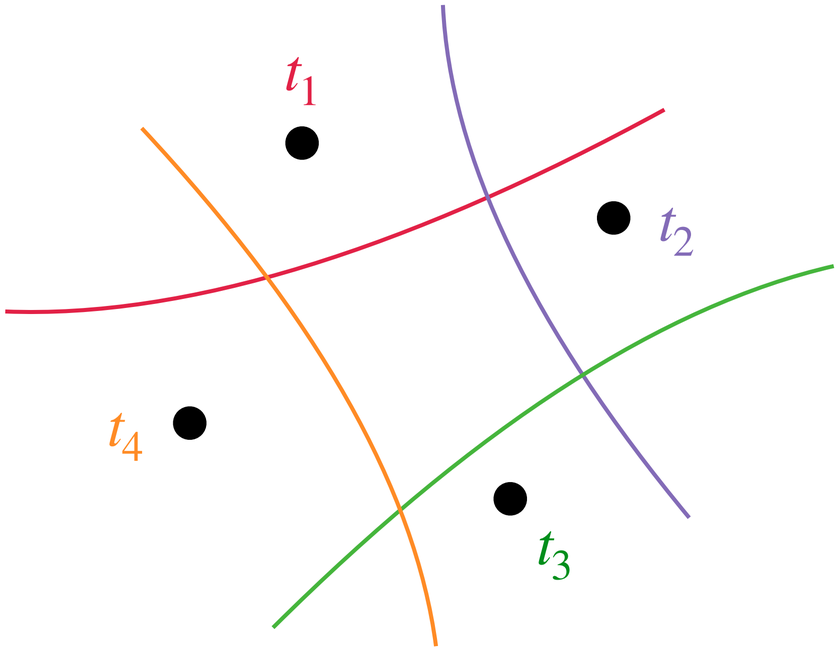
Even the simplest algorithm is based on a novel concept of **important cuts**.

Important Cuts

Daniel Marx

MULTIWAYCUT (MWC)

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:

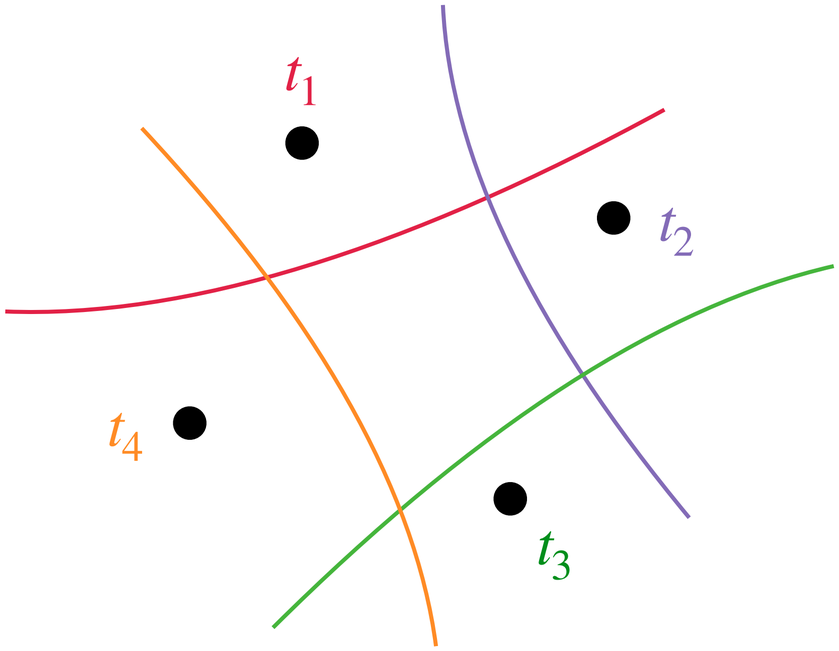
- $t_1 - T \backslash t_1$ minimal cut
- $t_2 - T \backslash t_2$ minimal cut
- $t_3 - T \backslash t_3$ minimal cut
- $t_4 - T \backslash t_4$ minimal cut

Important Cuts

Daniel Marx

MULTIWAYCUT (MWC)

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

Branch

Important Cuts

Daniel Marx

MULTIWAYCUT (MWC)

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 $T \setminus t_i$, then **enumerate every minimal** $(t_i, T \setminus t_i)$**-cut** of size at most k, and branching of choosing one such cut S in the solution.
3. Set $G := G \setminus S$ and $k := k - |S|$.
4. Go to Step 1.

Important Cuts

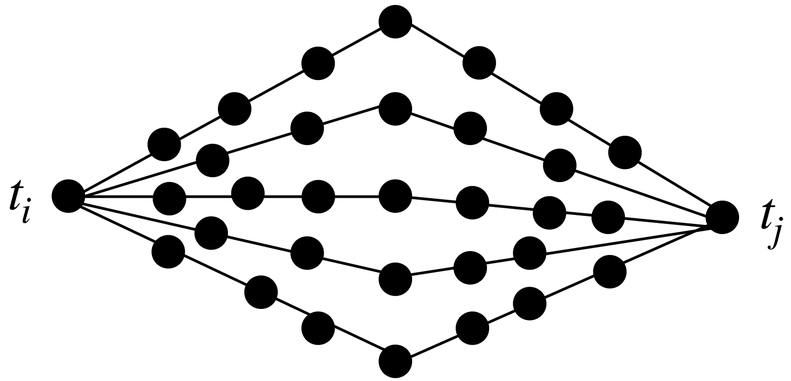
Daniel Marx

MULTIWAYCUT (MWC)

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

Naive algorithm

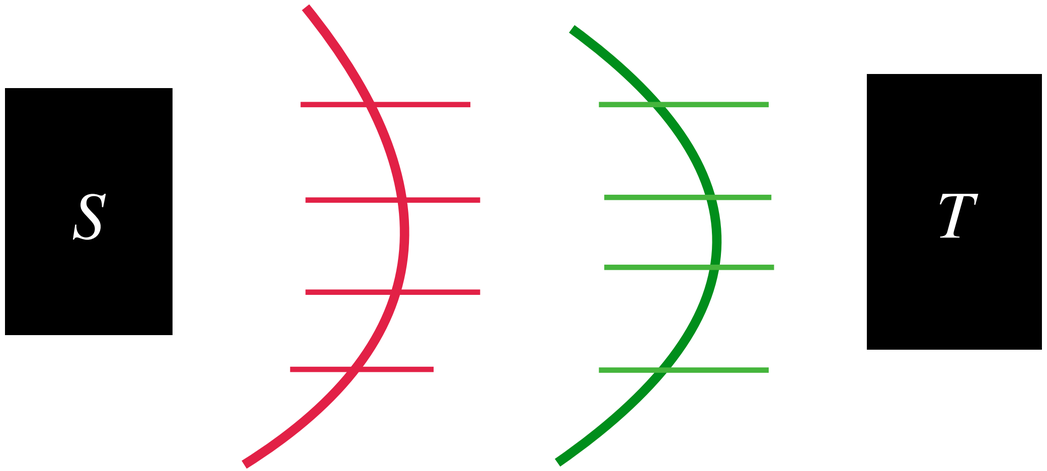
1. If each $t_i \in T$ is in a different connected component of G, then we are done.
2. If there exists $t_i \in T$, such that there is a path from t_i to $T \setminus t_i$, then **enumerate every minimal** $(t_i, T \setminus t_i)$**-cut** of size at most k, and branching of choosing one such cut S in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



Important Cuts: Directed and Undirected

Daniel Marx

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

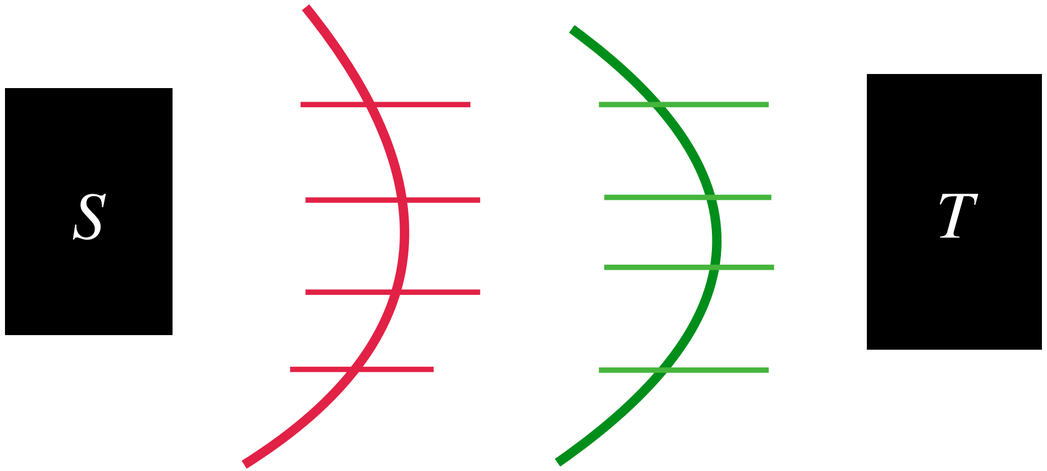


Green is better than red.

Important Cuts: Directed and Undirected

Daniel Marx

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



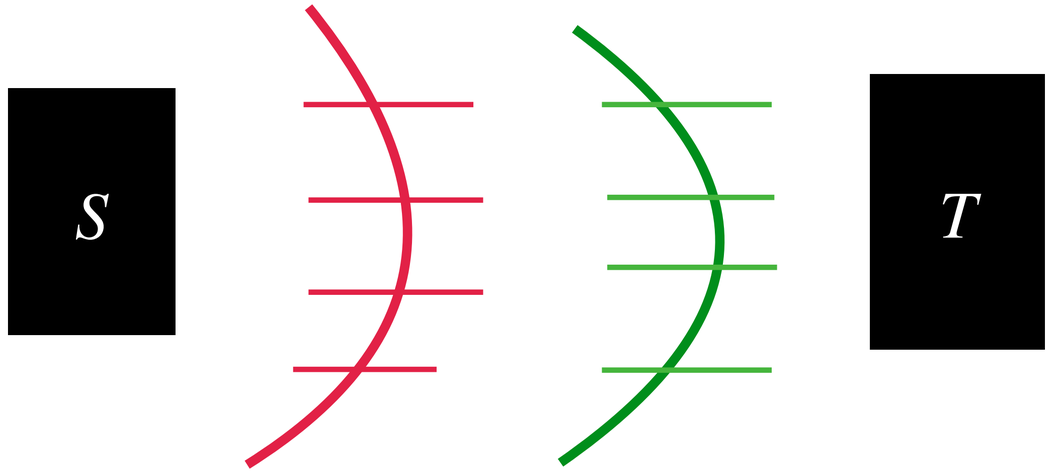
Green is better than red.

He called the maximum elements of this partial order as **important S-T cuts.**

Important Cuts: Directed and Undirected

Daniel Marx

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



Green is better than red.

He called the maximum elements of this partial order as **important S-T cuts.**

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

Important Cuts

Daniel Marx

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 $T \setminus t_i$, then **enumerate every important $(t_i, T \setminus t_i)$-cut** of size almost k, and branching of choosing one such cut S in the solution.
3. Set $G := G \setminus S$ and $k := k - |S|$.
4. Go to Step 1.

Important Cuts

Daniel Marx

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 $T \setminus t_i$, then **enumerate every important $(t_i, T \setminus t_i)$-cut** of size almost k, and branching of choosing one such cut S in the solution.
3. Set $G := G \setminus S$ and $k := k - |S|$.
4. Go to Step 1.

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

Important Cuts

Number of minimal incomparable S-T cuts (called **important S-T 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 minimal incomparable S-T cuts (called **important S-T 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.

Shadow Removal [STOC 2011]

~Random sampling of important cuts.

Daniel Marx Igor Razgon

The simple branching on important cuts does not work for example, for **MULTICUT**.

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.

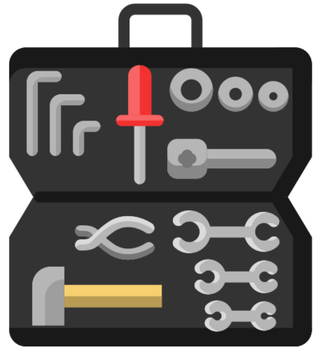
UNDIRECTED MULTICUT is FPT.

DIRECTED MULTIWAY is FPT.

Pre-pandemic~2016

Important Cuts

Shadow Removal

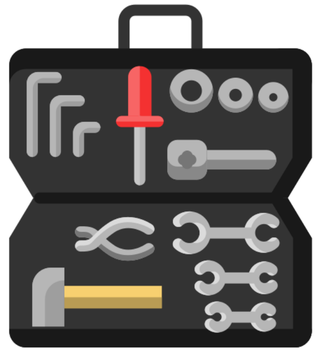


Other tools,
mostly for undirected problems

Pre-pandemic~2016

Important Cuts

Shadow Removal



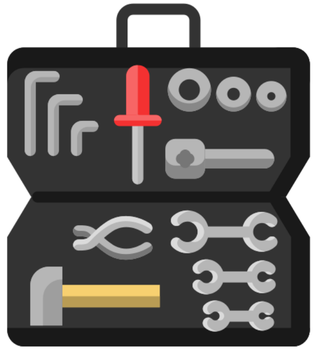
Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected				
Directed				

Pre-pandemic~2016

Important Cuts

Shadow Removal



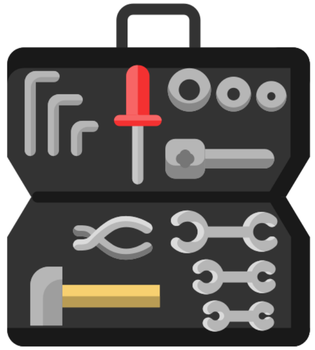
Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected				
Directed				

Pre-pandemic~2016

Important Cuts

Shadow Removal



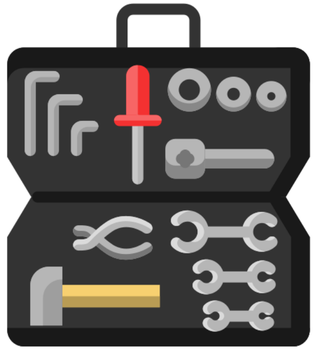
Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected			—	—
Directed				

Pre-pandemic~2016

Important Cuts

Shadow Removal



Other tools,
mostly for undirected problems

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected			—	—
Directed				

Pre-pandemic~2016

DIRECTED MULTICUT

Pre-pandemic~2016

DIRECTED MULTICUT

- W[1]-hard [Marx, Razgon STOC 2011, SICOMP 2014]

Pre-pandemic~2016

DIRECTED MULTICUT

- W[1]-hard [Marx, Razgon STOC 2011, SICOMP 2014]
- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]

Pre-pandemic~2016

DIRECTED MULTICUT

- W[1]-hard [Marx, Razgon STOC 2011, SICOMP 2014]
- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]
- W[1]-hard even with 4 terminal pairs [Pilipczuk, Wahlström SODA 2016, ACM TOCT 2018]

Pre-pandemic~2016

DIRECTED MULTICUT

- W[1]-hard [Marx, Razgon STOC 2011, SICOMP 2014]
- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]
- W[1]-hard even with 4 terminal pairs [Pilipczuk, Wahlström SODA 2016, ACM TOCT 2018]

Pre-pandemic~2016

DIRECTED MULTICUT

- W[1]-hard [Marx, Razgon STOC 2011, SICOMP 2014]
- FPT with 2 terminal pairs [Chitnis, Hajiaghayi, Marx SODA 2012, SICOMP 2013]
- W[1]-hard even with 4 terminal pairs [Pilipczuk, Wahlström SODA 2016, ACM TOCT 2018]

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?

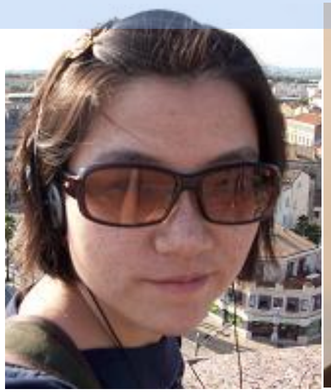
Find a solution of **size** at most k and **weight** at most W.
Parameter: k

III

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

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

Flow augmentation [STOC 2022]

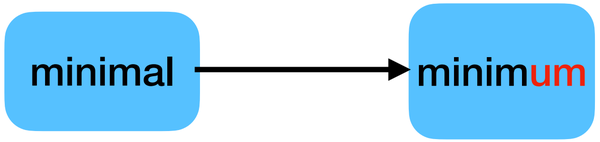


Eunjung Kim

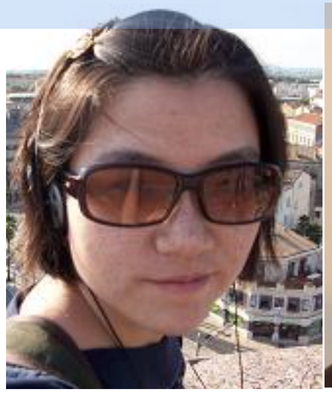
Stefan Krastch

Marcin Pilipczuk

Magnus Wahlström



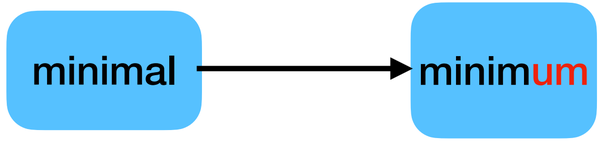
Flow augmentation [STOC 2022]


Eunjung Kim


Stefan Krastch

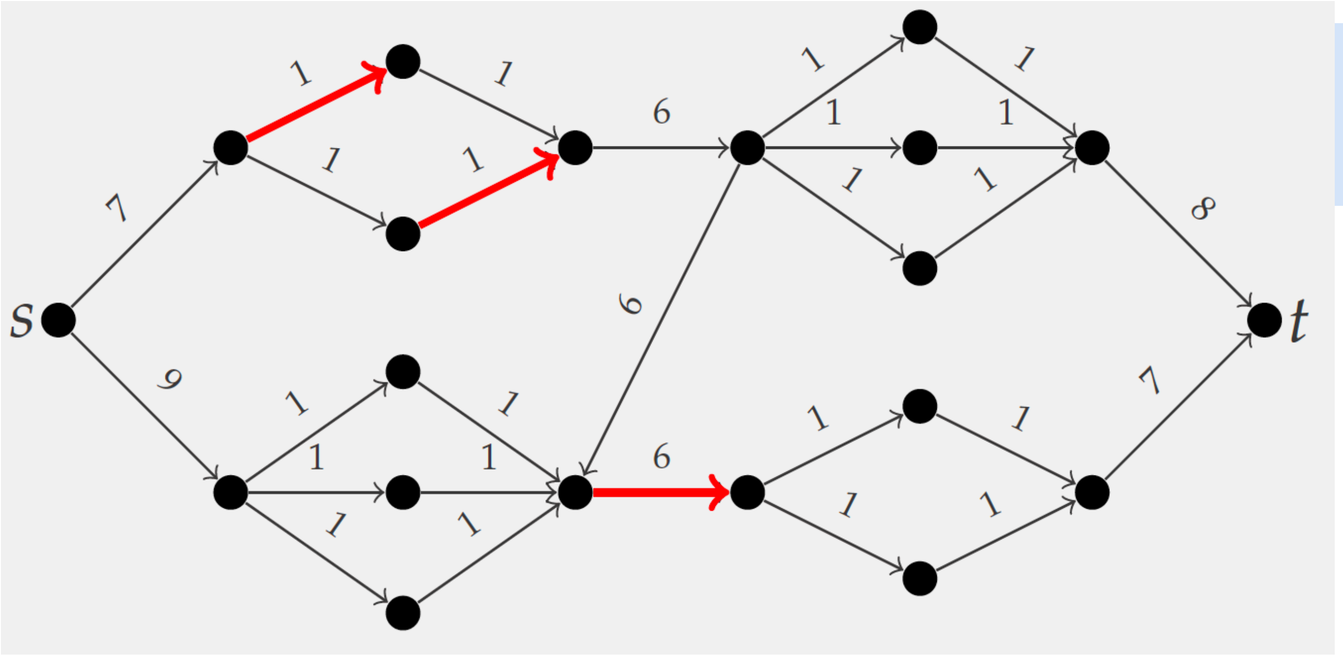

Marcin Pilipczuk


Magnus Wahlström



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

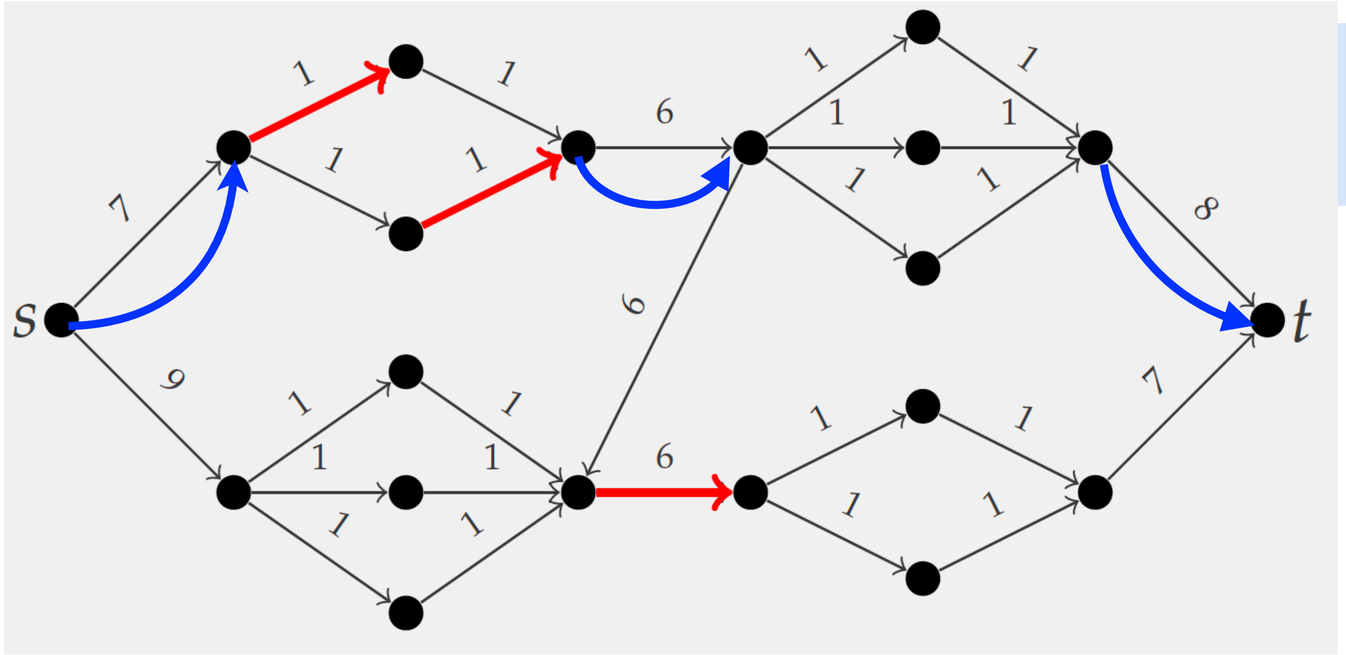
Guarantee: any **minimal** s-t separator Z of size at most k in G,
is a **minimum** s-t separator in G',
with probability $2^{-\mathcal{O}(k^4 \log k)}$



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 $2^{-\mathcal{O}(k^4 \log k)}$

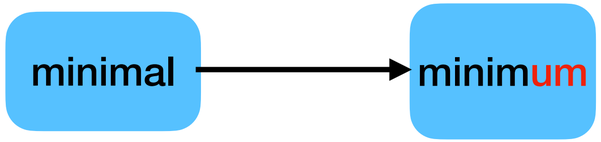


minimal → minimum

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,
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

Given a directed graph G,
find an s-t cut Z in G of size at most k and weight at most W.

(NP-hard)

Do flow-augmentation
and get G'.

Flow augmentation

minimal → minimum

Weighted s-t cut

Given a directed graph G,
find an s-t cut Z in G of size at most k and weight at most W.

(NP-hard)

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

Given a directed graph G,
find an s-t cut Z in G of size at most k and weight at most W.

(NP-hard)

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

Given a directed graph G,
find an s-t cut Z in G of size at most k and weight at most W.

(NP-hard)

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

Given a directed graph G,
find an s-t cut Z in G of size at most k and weight at most W.

(NP-hard)

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 $\lambda\tilde{\mathbf{W}} + \mathbf{W}$?

Questions?

Questions?

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?

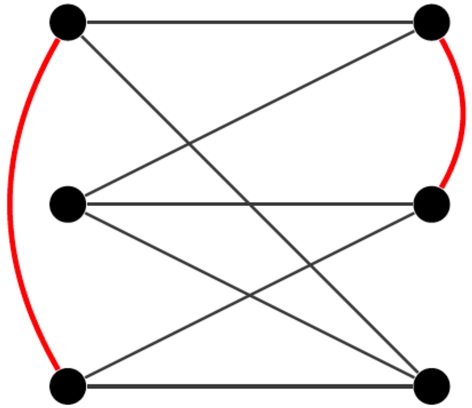
MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

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?

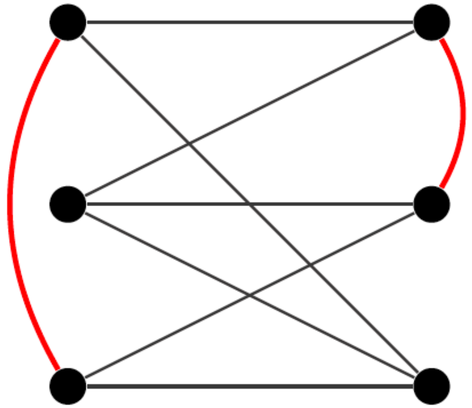
- $D = \{0,1\}, \Gamma = \{x \neq y\}$



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

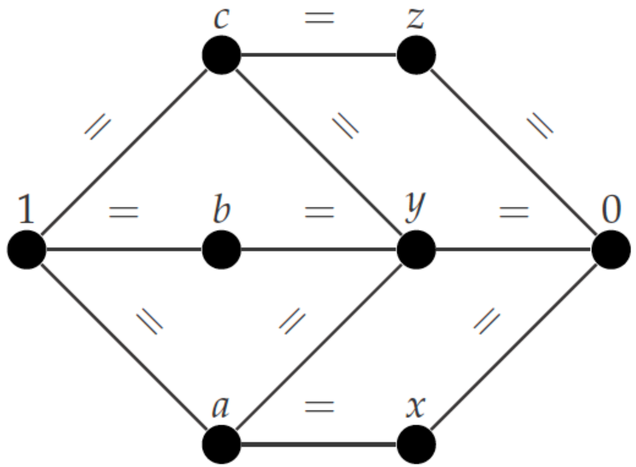
- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

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

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



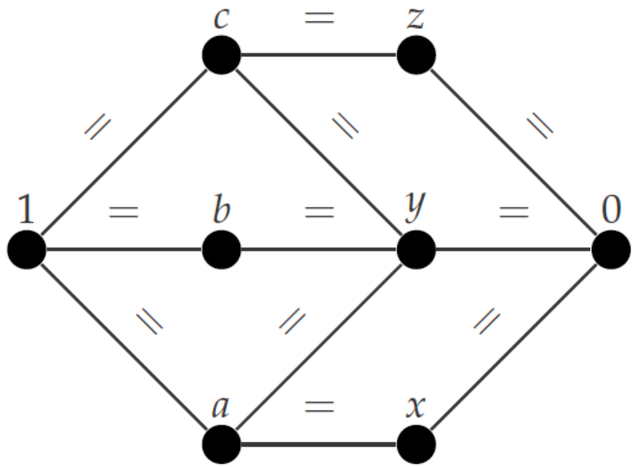
MinCSP (D, Γ): Can we delete at most k constraints to make the resulting instance satisfiable?

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

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?

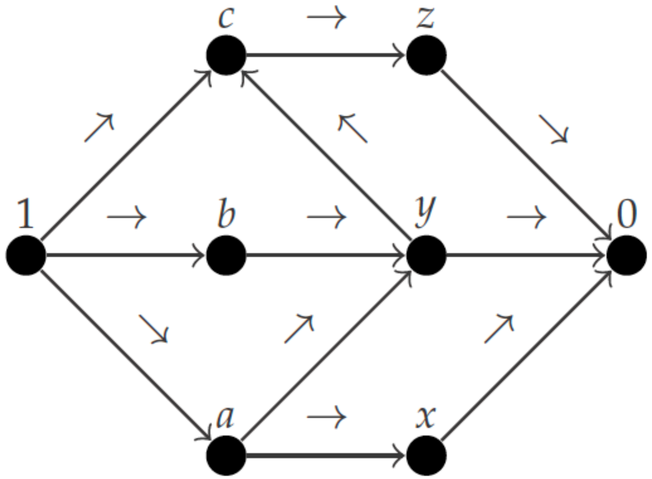
- $D = \{0,1\}, \Gamma = \{x \neq y\}$

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?

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

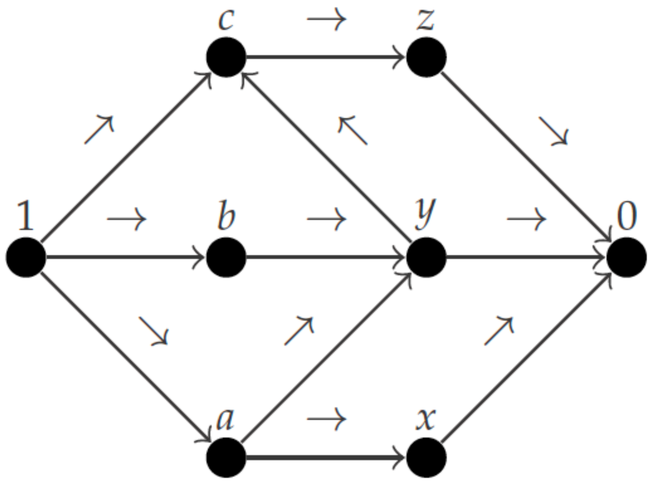
Edge Bipartization

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

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

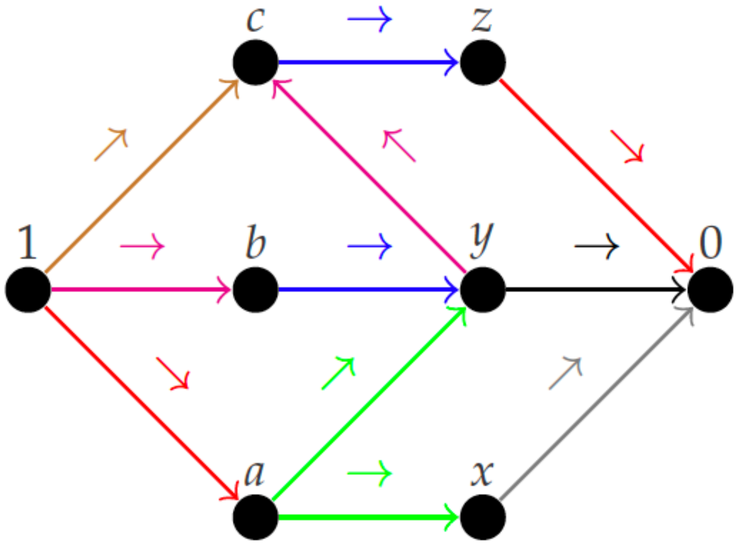
- $D = \{0,1\}, \Gamma = \{x = 1, x = 0, x = y\}$

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

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

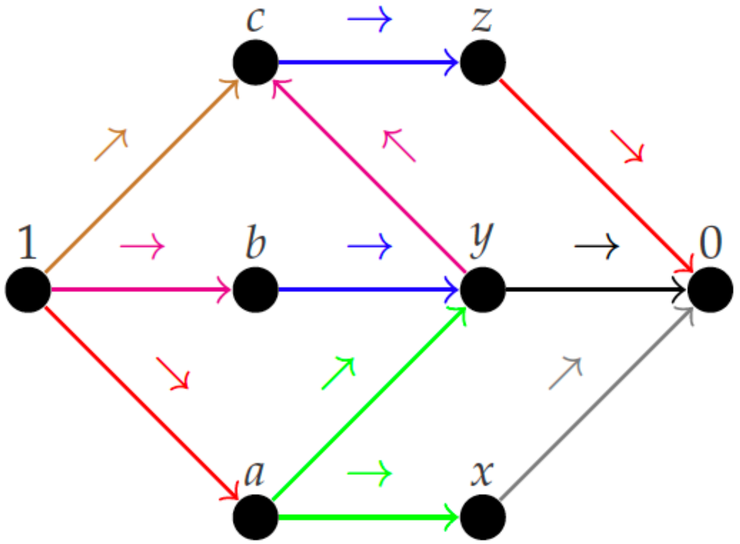
Undirected s-t cut

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

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

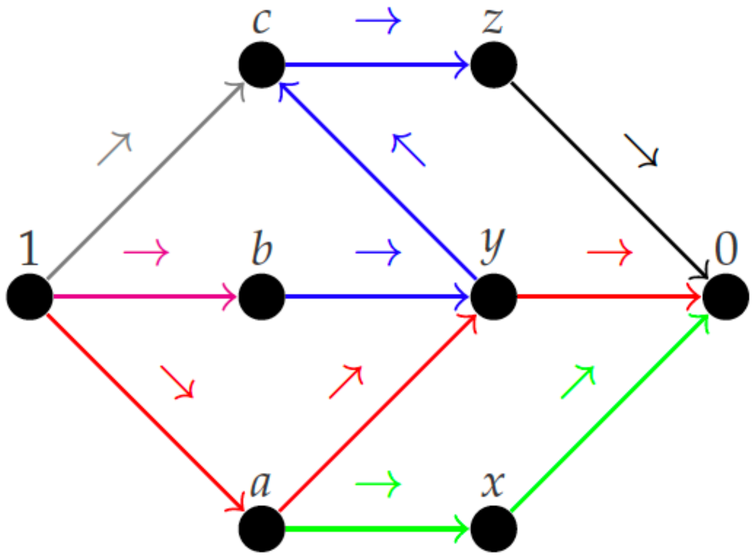
- $D = \{0,1\}, \Gamma = \{1 \to x, x \to 0, x \to y\}$

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

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

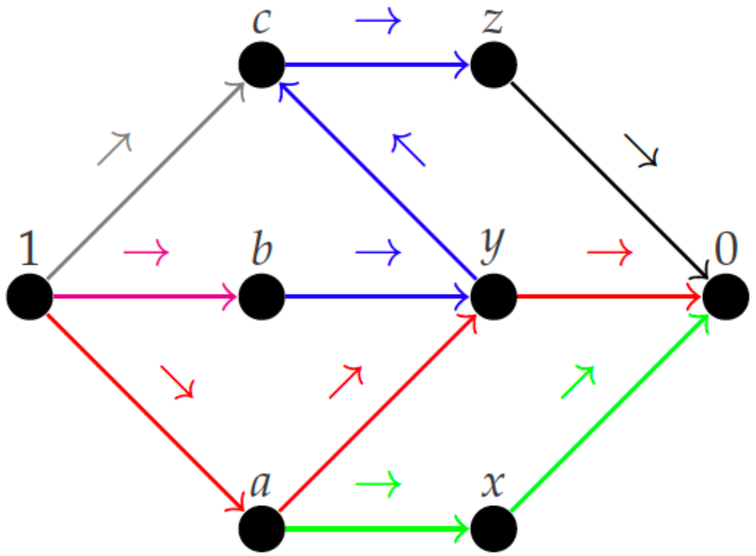
Directed s-t cut

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

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

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

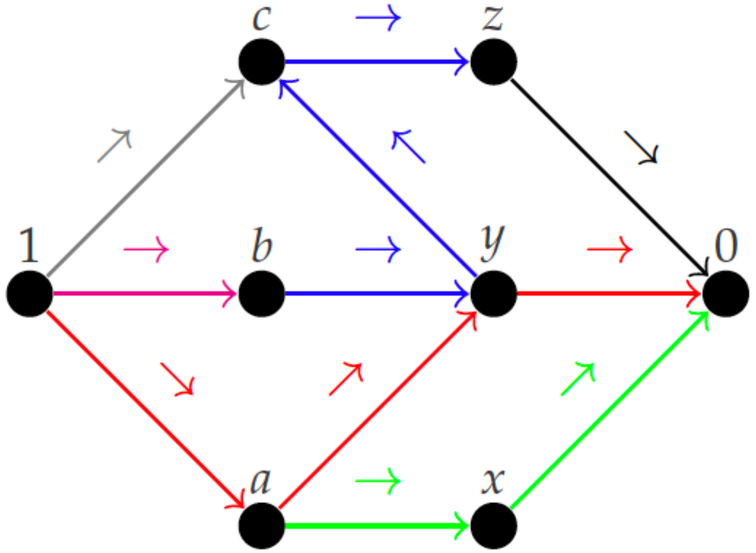
Directed s-t cut

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

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



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

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

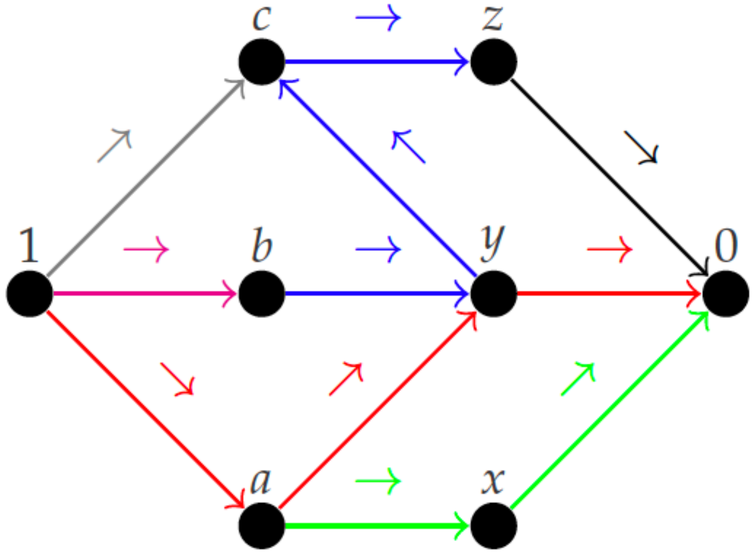
Directed s-t cut

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

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

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

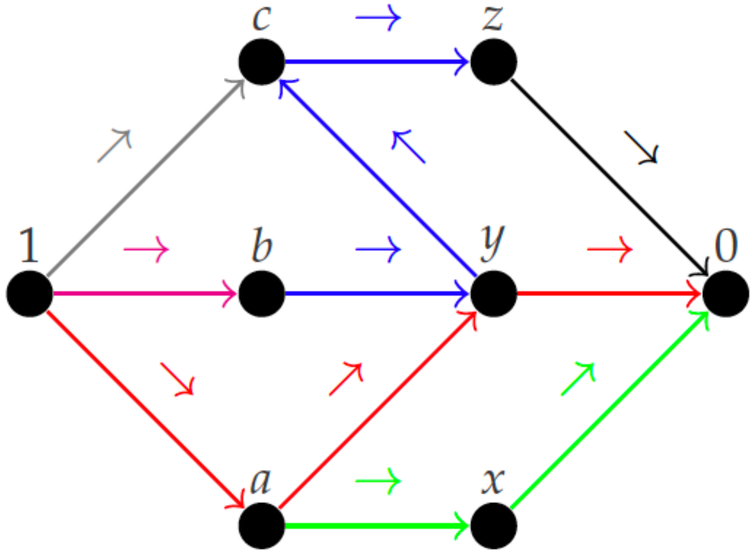
Directed s-t cut

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

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



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

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

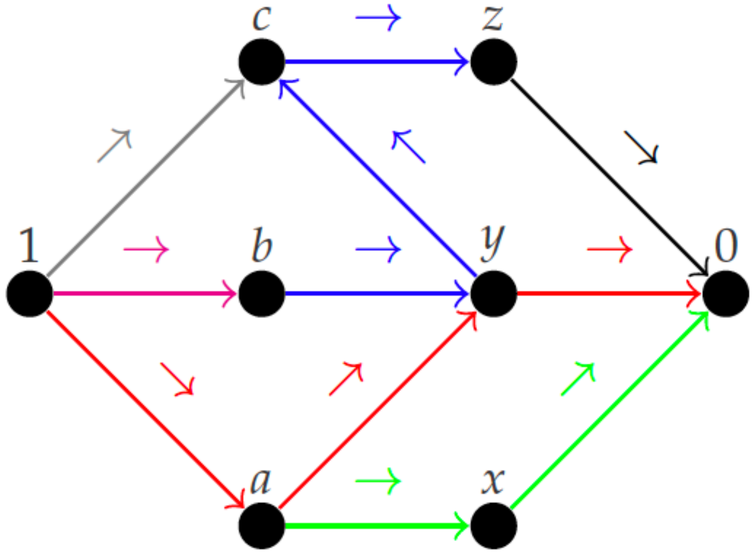
Directed s-t cut

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

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

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

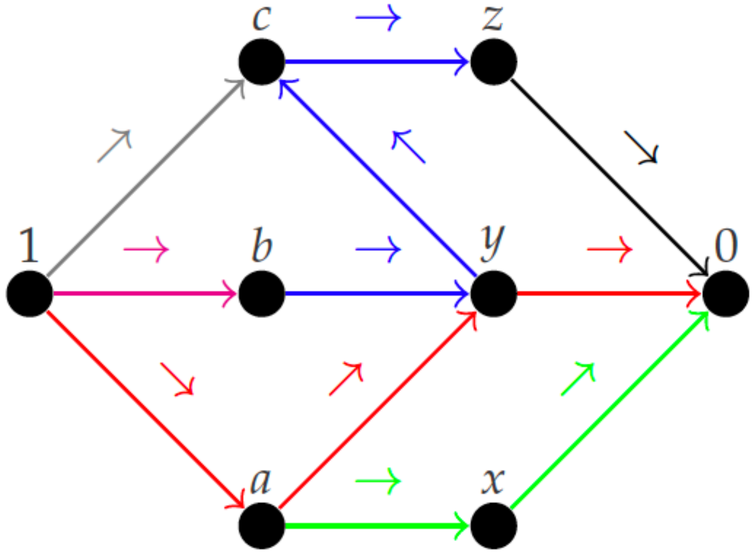
Directed s-t cut

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

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



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

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

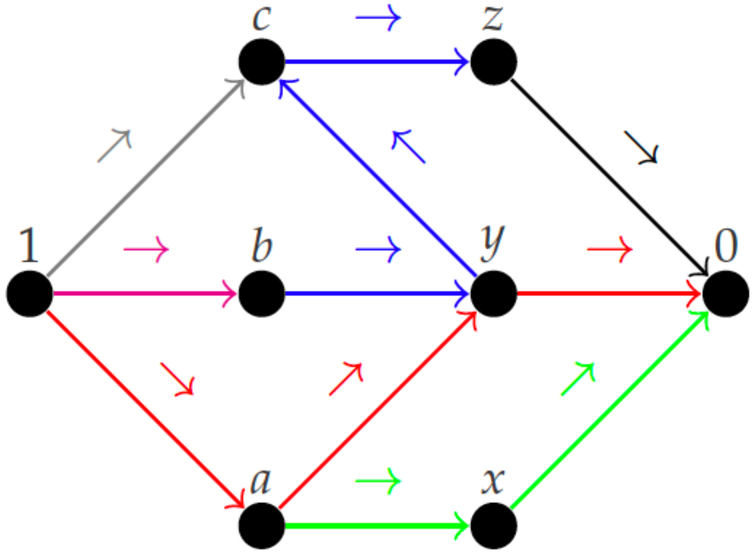
Directed s-t cut

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

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: Weighted MinCSP(Γ_{good})

- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- 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.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

FPT island: Weighted MinCSP(Γ_{good})

- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- 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.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

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 MinCSP(Γ_{good})

- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- 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.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

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 MinCSP(Γ_{good})

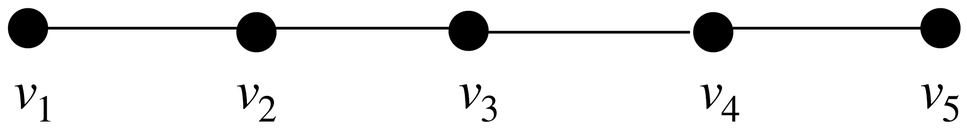
- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

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 MinCSP(Γ_{good})

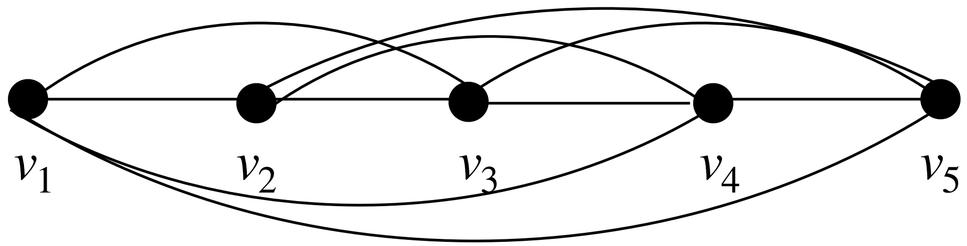
- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

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 MinCSP(Γ_{good})

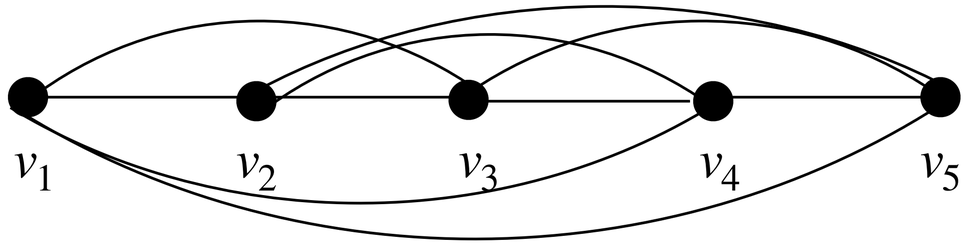
- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

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

FPT island: Weighted MinCSP(Γ_{good})

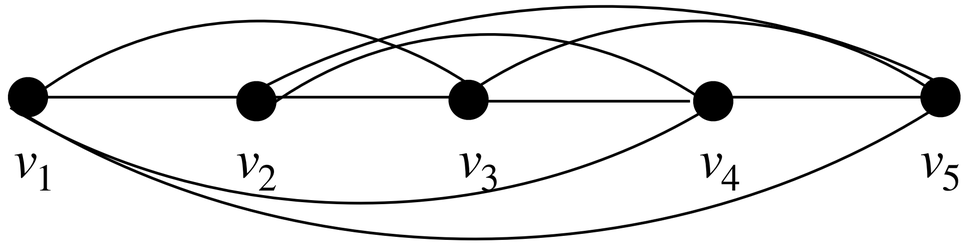
- Set of variables V
- Domain of variables is boolean $D = \{0,1\}$
- Collection of constraints:
  - Each constraint is an $\wedge$ of 2-ary clauses and clauses of the form $1 \to v$ or $v \to 0$.
  - The **constraint graph** is $2K_2$-free.
  - The **arity** of a constraint is the number of clauses in it.
- Each constraint has a weight.
- The goal is to find a satisfying assignment that satisfies all but at most k constraints of total weight at most W.

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

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**

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]

Weighted settings? [Kim, Masaryk, Pilipczuk, **Sharma**, Wahlström]

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected	 MinCSP $(=)$ (non-boolean)	 [KMP**S**W] MinCSP $(=, \neq)$ (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 $(<, \leq)$ (non-boolean)

Weighted settings? [Kim, Masaryk, Pilipczuk, **Sharma**, Wahlström]

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected	 MinCSP $(=)$ (non-boolean)	 [KMP**S**W] MinCSP $(=, \neq)$ (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 $(<, \leq)$ (non-boolean)

Weighted settings? [Kim, Masaryk, Pilipczuk, **Sharma**, Wahlström]

	Multiway Cut	Multicut	DFAS	Subset DFAS
Undirected	 MinCSP $(=)$ (non-boolean)	 [KMP**S**W] MinCSP $(=, \neq)$ (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 $(<, \leq)$ (non-boolean)

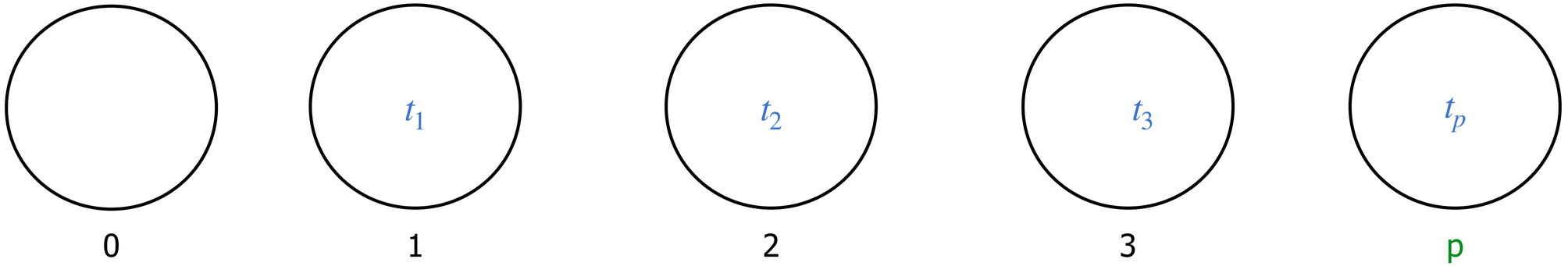
Undirected Multiway Cut

$\Rightarrow$ MinCSP (=)

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

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

G-Z (Z is a solution)



connected components of G-Z

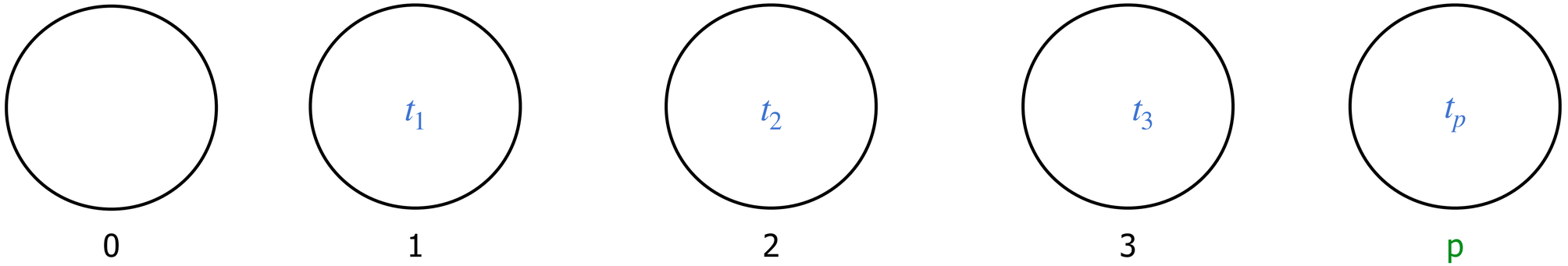
Undirected Multiway Cut

$\Longrightarrow$ MinCSP (=)

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

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

G-Z (Z is a solution)



connected components of G-Z

For every vertex $v \in V(G)$,	create a variable v	domain is $0,1,\ldots,p$
For every edge $uv \in E(G)$,	create a constraint $\mathbf{u = v}$	weight=w(uv)
For every $t_i \in T$,	create an undeletable constraint $\mathbf{t_i = i}$	

Encode this

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

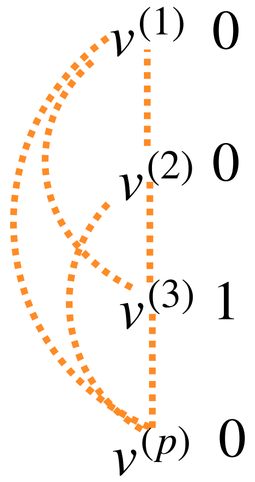
as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in binary

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

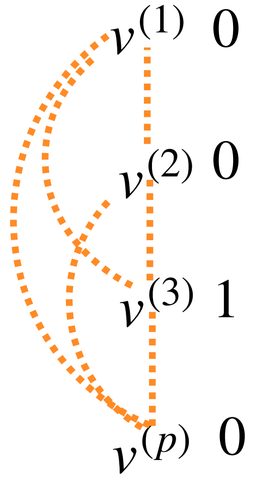
as MinCSP(Γ_{good})

Encoding bucket numbers (domain) in binary

- 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.

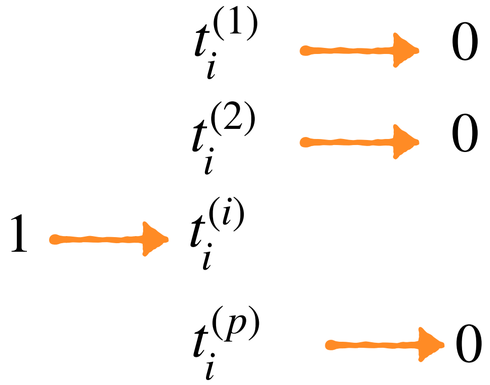
Encode this

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

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.



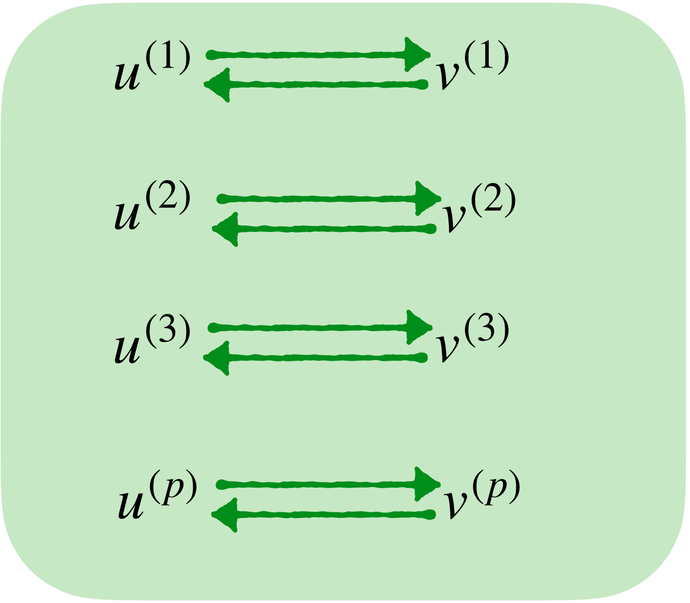
Encode this

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

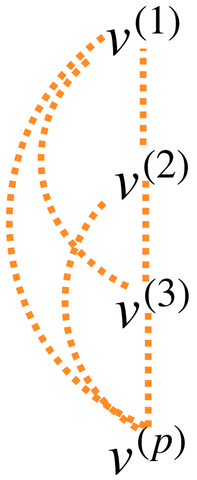
Domain $\{1,\ldots,p\}$

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

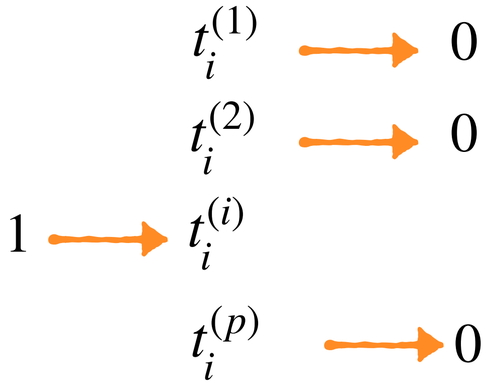
$t_i \in T$ $\quad \mathbf{t_i = i}$

as MinCSP(Γ_{good})

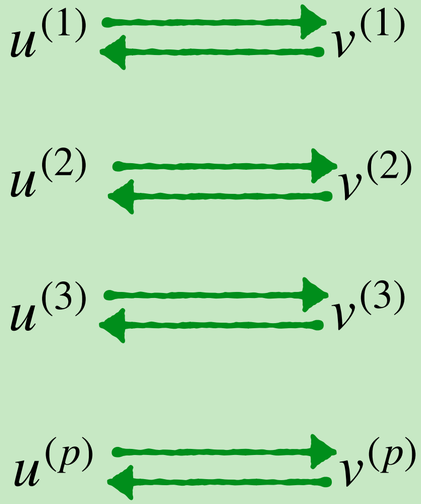
Encoding bucket numbers (domain) in binary



Forcing the vertices of T in the correct bucket



Forcing an edge uv in the same bucket



Encode this

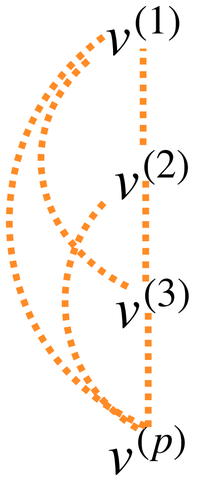
Domain $\{1,\ldots,p\}$

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

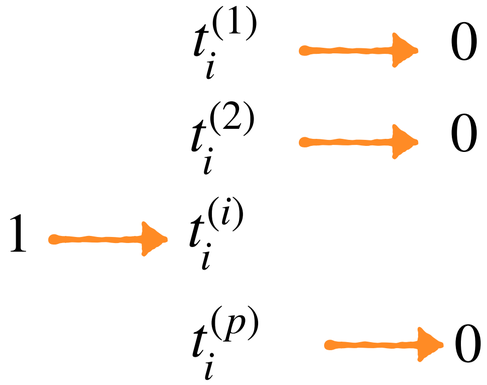
$t_i \in T$ $\quad \mathbf{t_i = i}$

as MinCSP(Γ_{good})

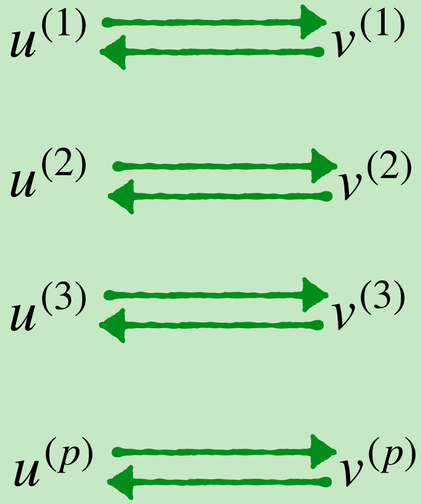
Encoding bucket numbers (domain) in binary



Forcing the vertices of T in the correct bucket



Forcing an edge uv in the same bucket



Encode this

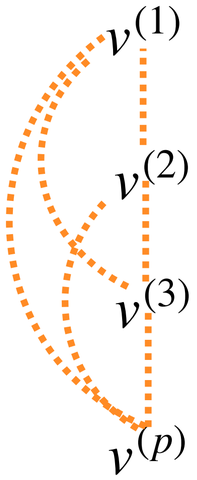
Domain $\{1,\ldots,p\}$

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

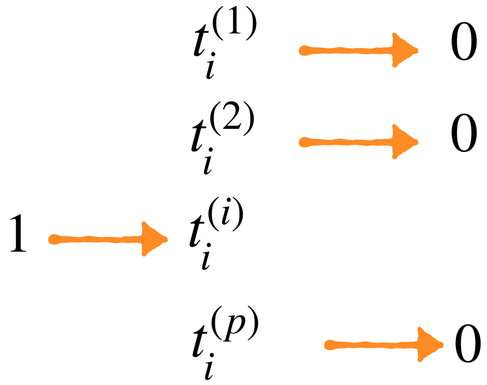
$t_i \in T$ $\quad \mathbf{t_i = i}$

as MinCSP(Γ_{good})

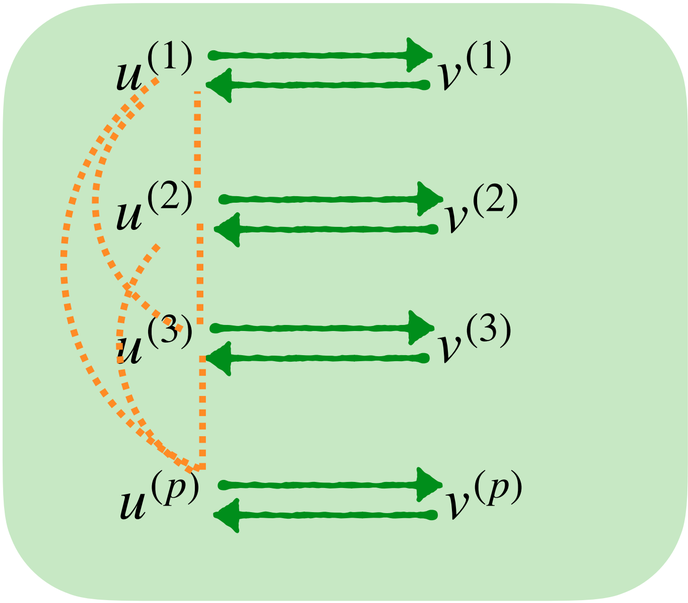
Encoding bucket numbers (domain) in binary



Forcing the vertices of T in the correct bucket



Forcing an edge uv in the same bucket



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~2019: 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

Beyond Boolean MinCSP

Min CSP
over Point Algebra

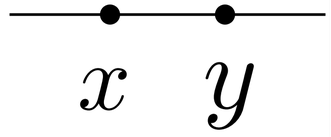
Point Algebra($=$, $\neq$, $<$, $\leq$)

Domain $\mathbb{Q}$

Constraints have access to

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

and are FO formulae

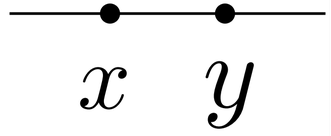


Min CSP
over Point Algebra

≡

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

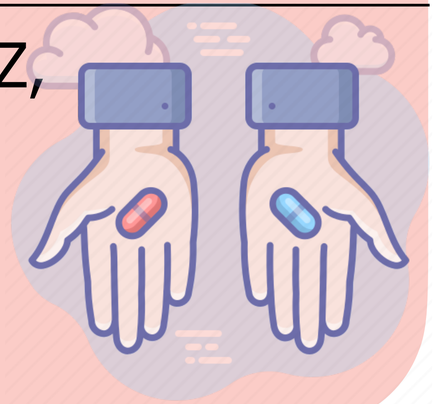
Constraints have access to
$<$, $=$, $\leq$, $\neq$
and are FO formulae



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.

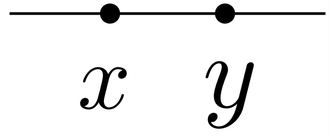


MIN CSP
over Point Algebra

≡

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

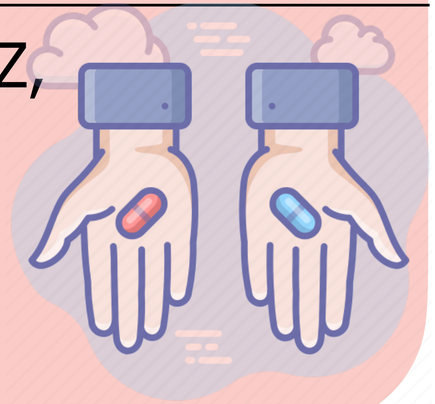
Constraints have access to
$<$, $=$, $\leq$, $\neq$
and are FO formulae



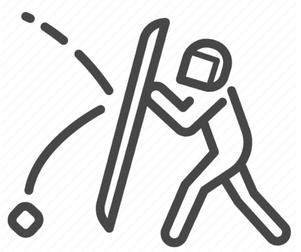
DIRECTED SYMMETRIC MULTICUT

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**]
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

Greedy tools (directed)

Important cuts

With flow-augmentation

Beyond flow-augmentation?

Summary

Before flow-augmentation

Greedy tools (directed)

Important cuts

With flow-augmentation

DIRECTED MULTICUT

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

Beyond flow-augmentation?

Summary

Before flow-augmentation

Greedy tools (directed)

Important cuts

With flow-augmentation

DIRECTED MULTICUT

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

Beyond flow-augmentation?

MIN CSP
over Point Algebra

Summary

Before flow-augmentation

Greedy tools (directed)

Important cuts

With flow-augmentation

DIRECTED MULTICUT

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

Beyond flow-augmentation?

MIN CSP
over Point Algebra

Summary

Before flow-augmentation

Greedy tools (directed)

Important cuts

With flow-augmentation

DIRECTED MULTICUT

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

Beyond flow-augmentation?

MIN CSP
over Point Algebra