CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Parameterized Complexity of Weighted Multicut in Trees

WG 2022

Esther Galby[1] Dániel Marx[1] **Philipp Schepper[1]**
Roohani Sharma[2] Prafullkumar Tale[1]

[1] CISPA Helmholtz Center for Information Security, Germany
[2] Max Planck Institute for Informatics, Saarland Informatics Campus, Germany

June 23, 2022

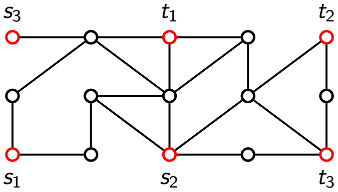
Definition of Multicut

Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

Multicut

Definition of Multicut

Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

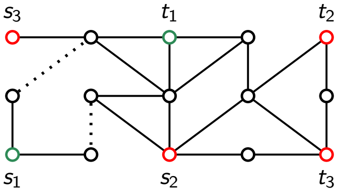


Multicut

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Definition of Multicut

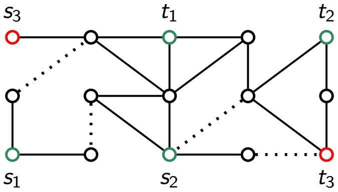
Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.



Multicut

Definition of Multicut

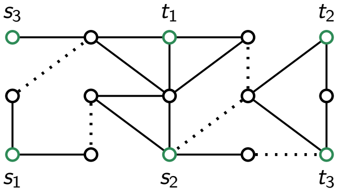
Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.



Multicut

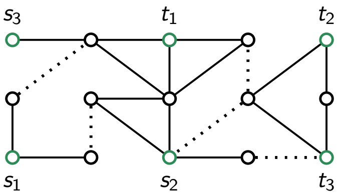
Definition of Multicut

Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.



Definition of Multicut

Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

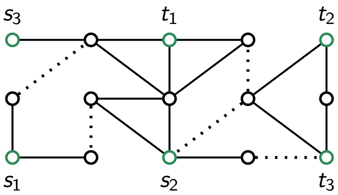


- $p = 1$: classical (s, t)-cut problem, poly-time solvable (Ford, Fulkerson ’62)

Multicut

Definition of Multicut

Input: An undirected graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

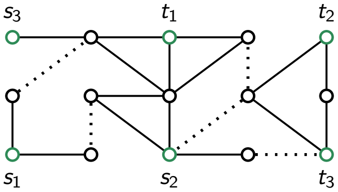


- $p = 1$: classical (s,t)-cut problem, poly-time solvable (Ford, Fulkerson '62)
- $p = 2$: Solvable in poly-time (Yannakakis et al. '83)

Multicut

Definition of Multicut

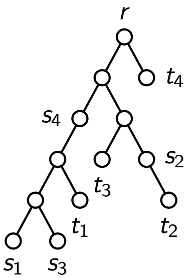
Input: An undirected graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$, and a positive integer k.
Question: Is there an edge set $S \subseteq E(G)$ with $|S| \leq k$, such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.



- $p = 1$: classical (s, t)-cut problem, poly-time solvable (Ford, Fulkerson '62)
- $p = 2$: Solvable in poly-time (Yannakakis et al. '83)
- $p = 3$: NP-hard (Dahlhaus et al. '94)

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

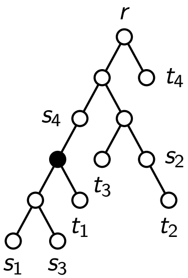


Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that:
   the lowest common ancestor v is farthest from r.

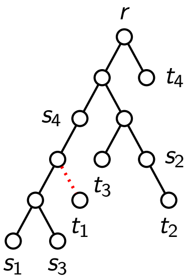


Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that: the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.

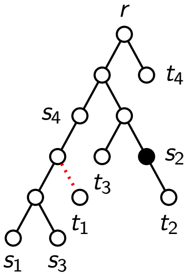


Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that: the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.

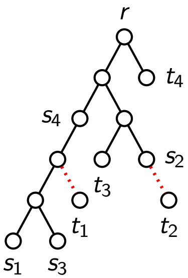


Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that: the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.

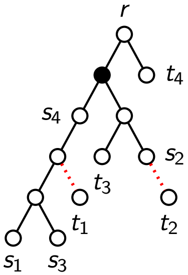


Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that: the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.

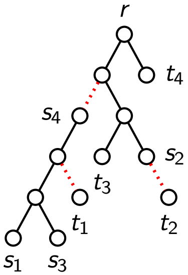


Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that:
   the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.



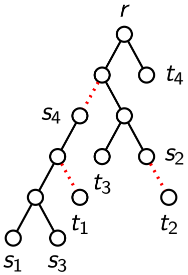
Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that:
   the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.

Running time: $2^k n^{\mathcal{O}(1)}$



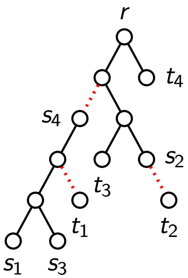
Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that: the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.

Running time: $2^k n^{\mathcal{O}(1)}$



General graphs:
$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$ algorithm based on important separators (Marx and Razgon '14, Bousquet et al. '18)

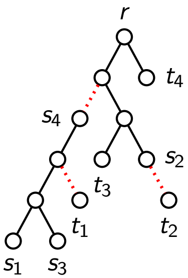
Trees as Input

Simple branching algorithm (Guo and Niedermeier '05):

Let r be the root of the tree.

1. Pick the connected pair (s_i, t_i) such that: the lowest common ancestor v is farthest from r.
2. Guess which of the two "outgoing" edges of v are deleted.
3. Repeat.

Running time: $2^k n^{\mathcal{O}(1)}$



General graphs:
$2^{\mathcal{O}(k^3)} n^{\mathcal{O}(1)}$ algorithm based on important separators
(Marx and Razgon '14, Bousquet et al. '18)

A problem with running time $f(k) \cdot n^{\mathcal{O}(1)}$ is *fixed parameter tractable* (FPT).
FPT also denotes the class of "efficient" problems in the parameterized setting.

Adding Weights

As for other problems keep size constraint and add weight constraint.

- Weighted (s, t)-Cut
- Weighted Directed Feedback Vertex Set
- Weighted Steiner Tree

Adding Weights

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

As for other problems keep size constraint and add weight constraint.

- Weighted (s, t)-Cut
- Weighted Directed Feedback Vertex Set
- Weighted Steiner Tree

Weighted Multicut (wMC)

Input: An undirected graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$,
a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W,
and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$
such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

Adding Weights

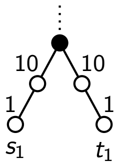
As for other problems keep size constraint and add weight constraint.

- Weighted (s, t)-Cut
- Weighted Directed Feedback Vertex Set
- Weighted Steiner Tree

Weighted Multicut (wMC)

Input: An undirected graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$ such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

All previous algorithms fail to generalize!



Adding Weights

As for other problems keep size constraint and add weight constraint.

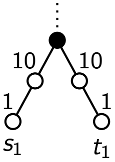
- Weighted (s, t)-Cut
- Weighted Directed Feedback Vertex Set
- Weighted Steiner Tree

Weighted Multicut (wMC)

Input: An undirected graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\text{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \text{wt}(e) \leq W$ such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

All previous algorithms fail to generalize!

Goal: Solve more restrictive versions first



Adding Weights

As for other problems keep size constraint and add weight constraint.

- Weighted (s, t)-Cut
- Weighted Directed Feedback Vertex Set
- Weighted Steiner Tree

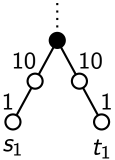
Weighted Multicut (wMC)

Input: An undirected graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$ such that for all $i \in [p]$: there is no path from s_i to t_i in $G - S$.

All previous algorithms fail to generalize!

Goal: Solve more restrictive versions first

$\implies$ Focus on (subdivided) stars



Hardness on Stars

(Subdivided) Stars seem to be important to handle:

Hardness on Stars

(Subdivided) Stars seem to be important to handle:

Theorem (Garg et al. '97)

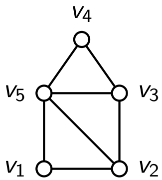
(Weighted) Multicut is NP-hard on stars.

(Subdivided) Stars seem to be important to handle:

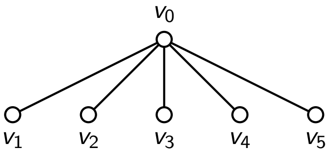
Theorem (Garg et al. '97)

(Weighted) Multicut is NP-hard on stars.

Vertex Cover



Multicut on Stars



(v_1, v_2)
(v_2, v_3)
(v_3, v_4)
(v_4, v_5)
(v_2, v_5)
(v_3, v_5)

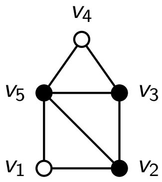
Hardness on Stars

(Subdivided) Stars seem to be important to handle:

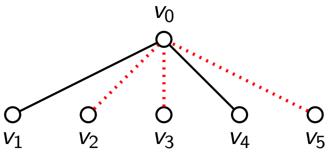
Theorem (Garg et al. '97)

(Weighted) Multicut is NP-hard on stars.

Vertex Cover



Multicut on Stars



(v_1, v_2)
(v_2, v_3)
(v_3, v_4)
(v_4, v_5)
(v_2, v_5)
(v_3, v_5)

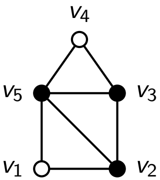
Hardness on Stars

(Subdivided) Stars seem to be important to handle:

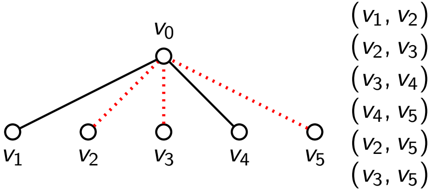
Theorem (Garg et al. '97)

(Weighted) Multicut is NP-hard on stars.

Vertex Cover



Multicut on Stars



(Weighted) Vertex Cover and (Weighted) Multicut on Stars are equivalent.

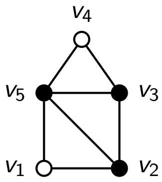
Hardness on Stars

(Subdivided) Stars seem to be important to handle:

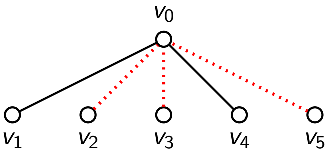
Theorem (Garg et al. '97)

(Weighted) Multicut is NP-hard on stars.

Vertex Cover



Multicut on Stars



(v_1, v_2)
(v_2, v_3)
(v_3, v_4)
(v_4, v_5)
(v_2, v_5)
(v_3, v_5)

(Weighted) Vertex Cover and (Weighted) Multicut on Stars are equivalent.

More evidence that (subdivided) stars are important.

Solving Bicriteria Problems

First criterion: size bound for solution, i.e. $|S| \leq k$

Solving Bicriteria Problems

First criterion: size bound for solution, i.e. $|S| \leq k$
Second criterion: "weight" of the solution, i.e. $\sum_{e \in S} \mathrm{wt}(e) \leq W$

Solving Bicriteria Problems

First criterion: size bound for solution, i.e. $|S| \leq k$
Second criterion: "weight" of the solution, i.e. $\sum_{e \in S} \mathrm{wt}(e) \leq W$

- Directed Feedback Vertex Set
- (s, t)-Cut
- Almost 2-SAT
- Digraph Pair-Cut

are solved in the unweighted setting but the weighted setting was long not known.

Solving Bicriteria Problems

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

First criterion: size bound for solution, i.e. $|S| \leq k$
Second criterion: "weight" of the solution, i.e. $\sum_{e \in S} \mathrm{wt}(e) \leq W$

- Directed Feedback Vertex Set
- (s, t)-Cut
- Almost 2-SAT
- Digraph Pair-Cut

are solved in the unweighted setting but the weighted setting was long not known.

Main issue: techniques for unweighted setting fail to generalize.

Solving Bicriteria Problems

First criterion: size bound for solution, i.e. $|S| \leq k$
Second criterion: "weight" of the solution, i.e. $\sum_{e \in S} \mathrm{wt}(e) \leq W$

- Directed Feedback Vertex Set
- (s, t)-Cut
- Almost 2-SAT
- Digraph Pair-Cut

are solved in the unweighted setting but the weighted setting was long not known.

Main issue: techniques for unweighted setting fail to generalize.

Theorem (Kim et al., STOC'22)

The weighted versions of these problems are (randomized) FPT.

The proof uses directed flow augmentation.

Weighted Digraph Pair-Cut

Definition

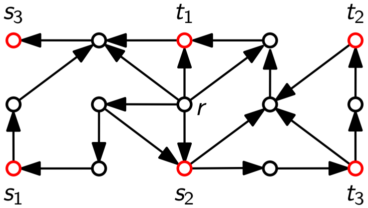
Input: A directed graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, a source vertex $r \in V(G)$, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$ such that for all $i \in [p]$:
if there is a path from r to s_i in $G - S$, then there is no path from r to t_i in $G - S$.

Weighted Digraph Pair-Cut

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Definition

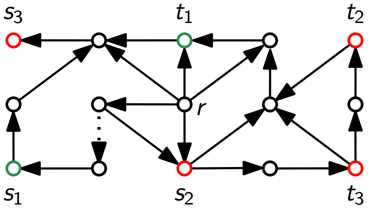
Input: A directed graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, a source vertex $r \in V(G)$, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$ such that for all $i \in [p]$:
if there is a path from r to s_i in $G - S$, then there is no path from r to t_i in $G - S$.



Weighted Digraph Pair-Cut

Definition

Input: A directed graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, a source vertex $r \in V(G)$, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$ such that for all $i \in [p]$:
if there is a path from r to s_i in $G - S$, then there is no path from r to t_i in $G - S$.

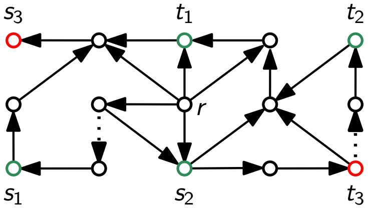


Weighted Digraph Pair-Cut

Definition

Input: A directed graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$, a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W, a source vertex $r \in V(G)$, and a positive integer k.

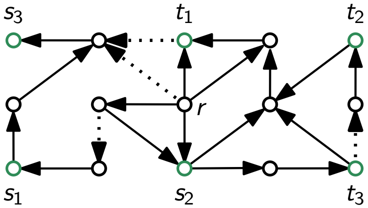
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$ such that for all $i \in [p]$:
if there is a path from r to s_i in $G - S$, then there is no path from r to t_i in $G - S$.



Weighted Digraph Pair-Cut

Definition

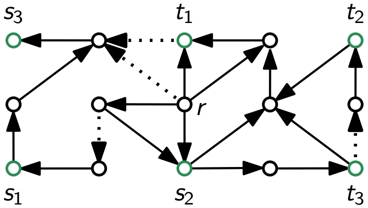
Input: A directed graph G, vertex pairs $(s_1, t_1), \dots, (s_p, t_p) \in V(G) \times V(G)$,
a weight function $\mathrm{wt} : E(G) \to \mathbb{N}$, an integer weight budget W,
a source vertex $r \in V(G)$, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \mathrm{wt}(e) \leq W$
such that for all $i \in [p]$:
if there is a path from r to s_i in $G - S$, then there is no path from r to t_i in $G - S$.



Weighted Digraph Pair-Cut

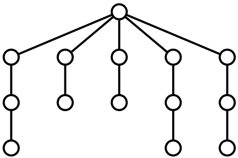
Definition

Input: A directed graph G, vertex pairs $(s_1, t_1), \ldots, (s_p, t_p) \in V(G) \times V(G)$,
a weight function $\text{wt} : E(G) \to \mathbb{N}$, an integer weight budget W,
a source vertex $r \in V(G)$, and a positive integer k.
Question: Is there a set $S \subseteq E(G)$ with $|S| \leq k$ and $\sum_{e \in S} \text{wt}(e) \leq W$
such that for all $i \in [p]$:
if there is a path from r to s_i in $G - S$, then there is no path from r to t_i in $G - S$.



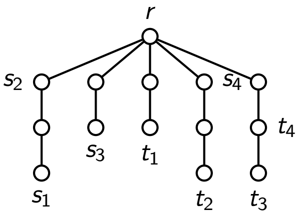
Recall: Weighted Digraph Pair-Cut is FPT.

Weighted Multicut on Subdivided Stars



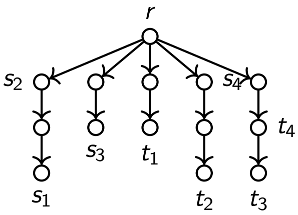
Weighted Multicut on Subdivided Stars

- Assume all pairs use the root r.



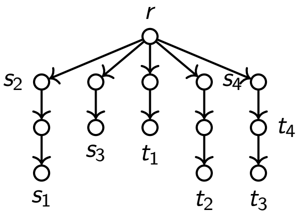
Weighted Multicut on Subdivided Stars

- Assume all pairs use the root r.
- Orient all edges away from r.



Weighted Multicut on Subdivided Stars

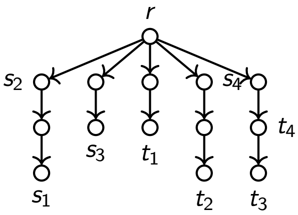
- Assume all pairs use the root r.
- Orient all edges away from r.
- Solve the constructed Weighted Digraph Pair-Cut instance.



Weighted Multicut on Subdivided Stars

- Assume all pairs use the root r.
- Orient all edges away from r.
- Solve the constructed Weighted Digraph Pair-Cut instance.

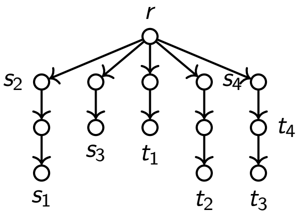
Observe: This also works for trees if we have the assumption!



Weighted Multicut on Subdivided Stars

- Assume all pairs use the root r.
- Orient all edges away from r.
- Solve the constructed Weighted Digraph Pair-Cut instance.

Observe: This also works for trees if we have the assumption!

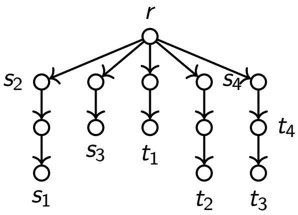


To achieve the assumption:
Compute an *unweighted* solution (with certain properties)
and then modify the graph while using the algorithm for subdivided stars.

Weighted Multicut on Subdivided Stars

- Assume all pairs use the root r.
- Orient all edges away from r.
- Solve the constructed Weighted Digraph Pair-Cut instance.

Observe: This also works for trees if we have the assumption!



To achieve the assumption:
Compute an *unweighted* solution (with certain properties)
and then modify the graph while using the algorithm for subdivided stars.

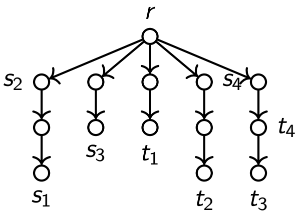
Main Theorem

Weighted Multicut on trees is FPT when parameterizing by the solution size k.

Weighted Multicut on Subdivided Stars

- Assume all pairs use the root r.
- Orient all edges away from r.
- Solve the constructed Weighted Digraph Pair-Cut instance.

Observe: This also works for trees if we have the assumption!



To achieve the assumption:
Compute an *unweighted* solution (with certain properties)
and then modify the graph while using the algorithm for subdivided stars.

Main Theorem

Weighted Multicut on trees is FPT when parameterizing by the solution size k.

Answers an implicit question by Bousquet et al. (STACS '09).

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Multicut	
Unweighted	FPT: Simple branching algorithm	
Weighted		

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Multicut	
Unweighted	FPT: Simple branching algorithm	
Weighted	FPT: Algorithm we have just seen	

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	
Weighted	FPT: Algorithm we have just seen	

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	poly-time: Greedily delete the lowest common ancestor
Weighted	FPT: Algorithm we have just seen	

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	poly-time: Greedily delete the lowest common ancestor
Weighted	FPT: Algorithm we have just seen	??

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	poly-time: Greedily delete the lowest common ancestor
Weighted	FPT: Algorithm we have just seen	??

Observation:
Weighted *Vertex* Multicut generalized Weighted *Edge* Multicut:

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	poly-time: Greedily delete the lowest common ancestor
Weighted	FPT: Algorithm we have just seen	??

Observation:
Weighted *Vertex* Multicut generalized Weighted *Edge* Multicut:

- Split each edge by a vertex which has the weight of the original edge.

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	poly-time: Greedily delete the lowest common ancestor
Weighted	FPT: Algorithm we have just seen	??

Observation:
Weighted *Vertex* Multicut generalized Weighted *Edge* Multicut:

- Split each edge by a vertex which has the weight of the original edge.
- Make all original vertices undeletable (e.g. infinite weight).

Edge Deletion vs. Vertex Deletion

Situation on trees:

	Edge Multicut	**Vertex Multicut**
Unweighted	FPT: Simple branching algorithm	poly-time: Greedily delete the lowest common ancestor
Weighted	FPT: Algorithm we have just seen	FPT: Algorithm we have just seen

Observation:
Weighted *Vertex* Multicut generalized Weighted *Edge* Multicut:

- Split each edge by a vertex which has the weight of the original edge.
- Make all original vertices undeletable (e.g. infinite weight).

Our algorithm also works for the vertex version!

Conclusion

We use results for Weighted Digraph Pair-Cut to show the following:

Main Theorem 1

Weighted Multicut on trees can be solved in randomized time $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$.

Conclusion

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

We use results for Weighted Digraph Pair-Cut to show the following:

Main Theorem 1

Weighted Multicut on trees can be solved in randomized time $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$.

Similarly we solve a version of the problem without the size constraint k.

Main Theorem 2

Weighted Multicut without size constraint on trees with ℓ leaves can be solved in time $2^{\mathcal{O}(\ell^3)} \cdot n^{\mathcal{O}(1)}$.

Conclusion

We use results for Weighted Digraph Pair-Cut to show the following:

Main Theorem 1

Weighted Multicut on trees can be solved in randomized time $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$.

Similarly we solve a version of the problem without the size constraint k.

Main Theorem 2

Weighted Multicut without size constraint on trees with ℓ leaves can be solved in time $2^{\mathcal{O}(\ell^3)} \cdot n^{\mathcal{O}(1)}$.

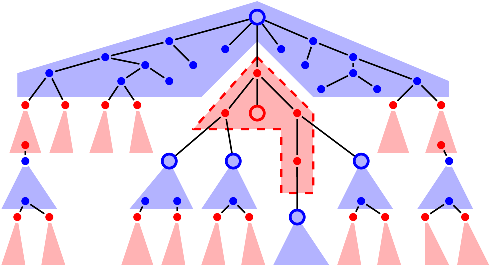
One more result generalizing Main Theorem 2 and a result by Guo and Niedermeier (2006) about request degree.

Full version: `arXiv:2205.10105`

Additional Material

(d, ℓ)-Light Instances

- Delete all vertices used for at most d terminal pair request.
- The closed neighborhood of the remaining components must has at most ℓ leaves.



Result for (d, ℓ)-Light Instances

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Parameter: request degree d and number of leaves ℓ

wMC on (d, ℓ)-light trees can be solved in time $3^d \cdot 2^{d\ell} \cdot 2^{\mathcal{O}(\ell^3)} \cdot n^{\mathcal{O}(1)}$
if we drop the size constraint.

Proof idea:

- For vertices with small ($\leq d$) request degree:
  Use dynamic programming.
- For components of vertices with large ($\geq d$) request degree:
  Use one of the new algorithms as subroutine as the component has at most ℓ leaves.

This implies a result by Guo and Niedermeier (2006) about the request degree d.

Parameterizing by Number of Leaves

Parameter: number of leaves ℓ

wMC on trees with ℓ leaves can be solved in time $2^{\mathcal{O}(\ell^3)} \cdot n^{\mathcal{O}(1)}$
if we drop the size constraint.

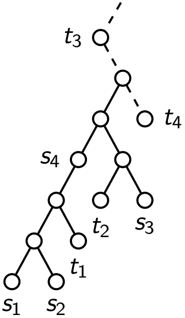
Proof idea:

- Use another result from Kim et al. '22 to solve the problem on paths and stars.
- Apply similar procedure as for previous algorithm to solve the problem on trees.

Algorithm – Main Idea

Preprocessing:

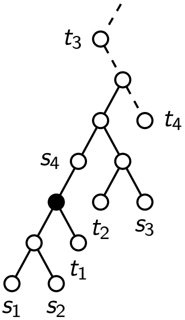
1 Compute a minimum *unweighted* solution X_{opt}.



Algorithm – Main Idea

Preprocessing:

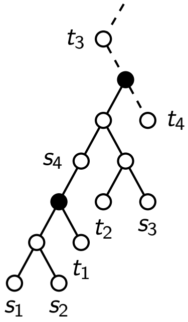
1. Compute a minimum *unweighted* solution X_{opt}.



Algorithm – Main Idea

Preprocessing:

1. Compute a minimum *unweighted* solution X_{opt}.
2. Extend X_{opt} to X by computing the closure under taking the "lowest common ancestor".



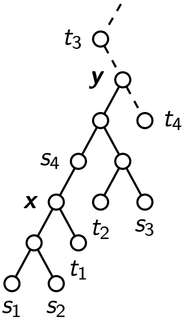
Algorithm – Main Idea

Preprocessing:

1. Compute a minimum *unweighted* solution X_{opt}.
2. Extend X_{opt} to X by computing the closure under taking the "lowest common ancestor".

Branching algorithm:

1. Pick $x \in X$ to be furthest from the root.
   Let $y \in X$ be its closest ancestor.



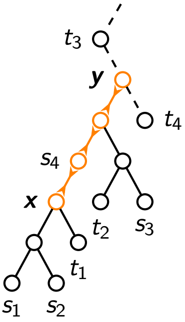
Algorithm – Main Idea

Preprocessing:

1. Compute a minimum *unweighted* solution X_{opt}.
2. Extend X_{opt} to X by computing the closure under taking the "lowest common ancestor".

Branching algorithm:

1. Pick $x \in X$ to be furthest from the root. Let $y \in X$ be its closest ancestor.
2. Guess if some vertex between x and y is selected.
3. Case "no such vertex": Contract the path from x to y onto an undeletable vertex.



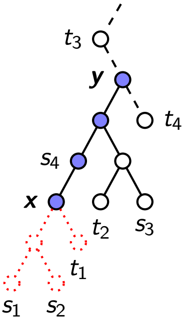
Algorithm – Main Idea

Preprocessing:

1. Compute a minimum *unweighted* solution X_{opt}.
2. Extend X_{opt} to X by computing the closure under taking the "lowest common ancestor".

Branching algorithm:

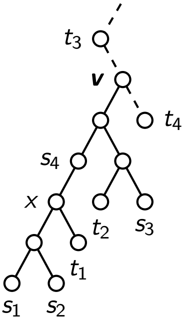
1. Pick $x \in X$ to be furthest from the root.
   Let $y \in X$ be its closest ancestor.
2. Guess if some vertex between x and y is selected.
3. Case "no such vertex":
   Contract the path from x to y onto an undeletable vertex.
4. Case "there is such a vertex":
   For each vertex v between x and y:
   Update $\text{wt}(v) = \text{wt}(v) + \text{OPT}(T^{\dagger}_{v,x})$ (next step)
   Delete $T^{\dagger}_{x}$ and add the pair (x, y).
5. Recurse.



Algorithm – Updating the Weights

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^{\dagger}_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.



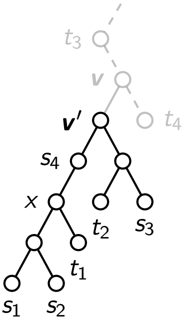
Algorithm – Updating the Weights

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^{\dagger}_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.

1. Consider the graph $T^{\dagger}_{v,x}$, i.e. the subtree of T_v containing x.
   Let v' be its root.
   Observe: For each $(s, t) \in \mathcal{P}|_{v'}$, the path from x to s or from x to t has to be cut.

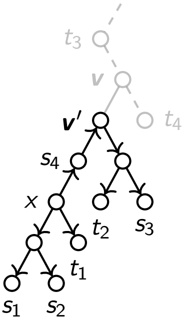


Algorithm – Updating the Weights

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^{\dagger}_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.

1. Consider the graph $T^{\dagger}_{v,x}$, i.e. the subtree of T_v containing x.
   Let v' be its root.
   Observe: For each $(s, t) \in \mathcal{P}|_{v'}$, the path from x to s or from x to t has to be cut.
2. Direct all edges away from x.
   Define the weight of each edge as the weight of its head.
   $\rightarrow$ Must solve a digraph pair-cut problem.

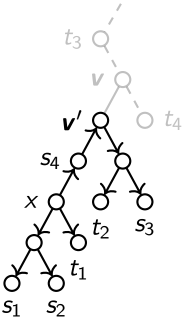


Algorithm – Updating the Weights

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^\dagger_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.

1. Consider the graph $T^\dagger_{v,x}$, i.e. the subtree of T_v containing x.
   Let v' be its root.
   Observe: For each $(s,t) \in \mathcal{P}|_{v'}$, the path from x to s or from x to t has to be cut.
2. Direct all edges away from x.
   Define the weight of each edge as the weight of its head.
   $\rightarrow$ Must solve a digraph pair-cut problem.
3. By Kim et al. '22 this can be solved in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$.
   Let $C_{v,i}$ be the optimal value.



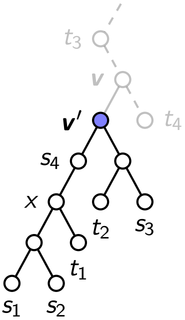
Algorithm – Updating the Weights

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^{\dagger}_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.

1. Consider the graph $T^{\dagger}_{v,x}$, i.e. the subtree of T_v containing x.
   Let v' be its root.
   Observe: For each $(s,t) \in \mathcal{P}|_{v'}$, the path from x to s or from x to t has to be cut.
2. Direct all edges away from x.
   Define the weight of each edge as the weight of its head.
   $\rightarrow$ Must solve a digraph pair-cut problem.
3. By Kim et al. '22 this can be solved in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$.
   Let $C_{v,i}$ be the optimal value.
4. Define $\mathrm{wt}(v) = \mathrm{wt}(v) + C_{v,i}$.

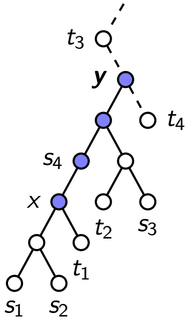


Algorithm – Updating the Weights

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^{\dagger}_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.

1. Consider the graph $T^{\dagger}_{v,x}$, i.e. the subtree of T_v containing x.
   Let v' be its root.
   Observe: For each $(s,t) \in \mathcal{P}|_{v'}$, the path from x to s or from x to t has to be cut.
2. Direct all edges away from x.
   Define the weight of each edge as the weight of its head.
   $\rightarrow$ Must solve a digraph pair-cut problem.
3. By Kim et al. '22 this can be solved in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$.
   Let $C_{v,i}$ be the optimal value.
4. Define $\mathrm{wt}(v) = \mathrm{wt}(v) + C_{v,i}$.



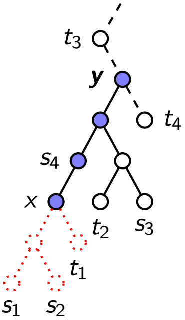
Repeat this for all vertices between x and y.

Algorithm – Updating the Weights

Goal: Compute for each v the optimal solution in the subtree below, i.e. in $T^{\dagger}_{v,x}$.

Guess the size $i \in [k]$ of the solution in this part.

1. Consider the graph $T^{\dagger}_{v,x}$, i.e. the subtree of T_v containing x.
   Let v' be its root.
   Observe: For each $(s,t) \in \mathcal{P}|_{v'}$, the path from x to s or from x to t has to be cut.
2. Direct all edges away from x.
   Define the weight of each edge as the weight of its head.
   $\rightarrow$ Must solve a digraph pair-cut problem.
3. By Kim et al. '22 this can be solved in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$.
   Let $C_{v,i}$ be the optimal value.
4. Define $\mathrm{wt}(v) = \mathrm{wt}(v) + C_{v,i}$.



Repeat this for all vertices between x and y.

Remove the subtree below x and the corresponding pairs.

Remove x from X and add (x, y) as a new pair to $\mathcal{P}$.

Algorithm – Running Time

Preprocessing:

- Solution X_{opt} and its closure X can be computed in polynomial time
- $|X| \leq 2|X_{\text{opt}}| \leq 2k$

Algorithm – Running Time

Preprocessing:

- Solution X_{opt} and its closure X can be computed in polynomial time
- $|X| \leq 2|X_{\text{opt}}| \leq 2k$

For each iteration of the **branching algorithm:**

- create $k+1$ new branches,
- create $\mathcal{O}(n)$ digraph pair-cut instances
- solve them in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$ due to the subroutine, and
- remove one vertex from X.

Algorithm – Running Time

Preprocessing:

- Solution X_{opt} and its closure X can be computed in polynomial time
- $|X| \leq 2|X_{\text{opt}}| \leq 2k$

For each iteration of the **branching algorithm:**

- create $k+1$ new branches,
- create $\mathcal{O}(n)$ digraph pair-cut instances
- solve them in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$ due to the subroutine, and
- remove one vertex from X.

$\implies$ Total running time is $k^{\mathcal{O}(k)} \cdot 2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$.

□