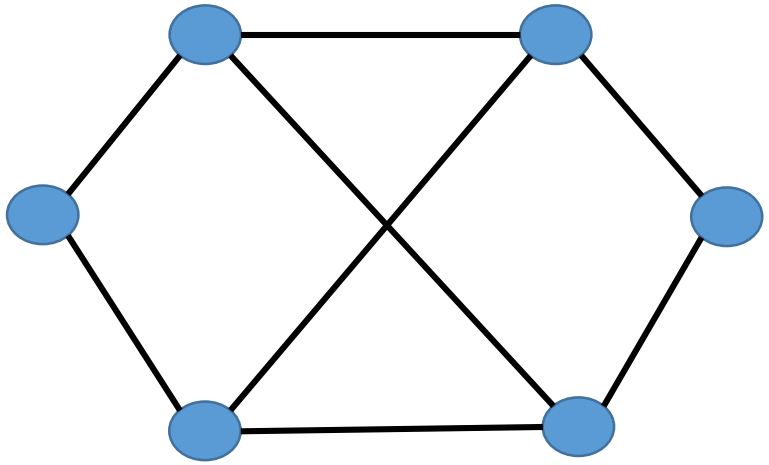
Improved Distributed Approximations for Minimum-Weight Two-Edge-Connected Spanning Subgraph

Michal Dory (Technion), Mohsen Ghaffari (ETH Zurich)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839

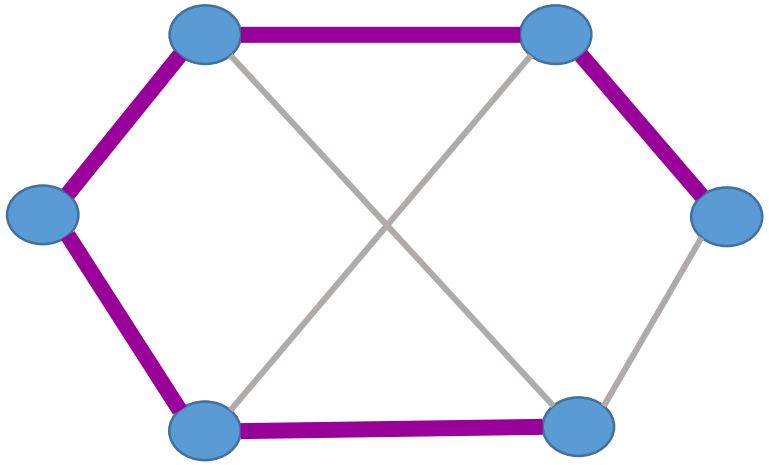
Network design

Goal: find a **low-cost subgraph** satisfying **nice** properties



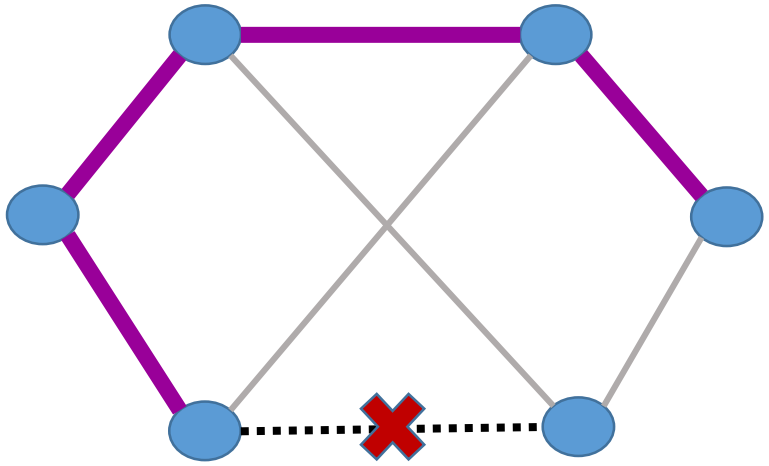
Minimum spanning tree (MST)

Goal: find a minimum weight **connected** subgraph



Minimum spanning tree (MST)

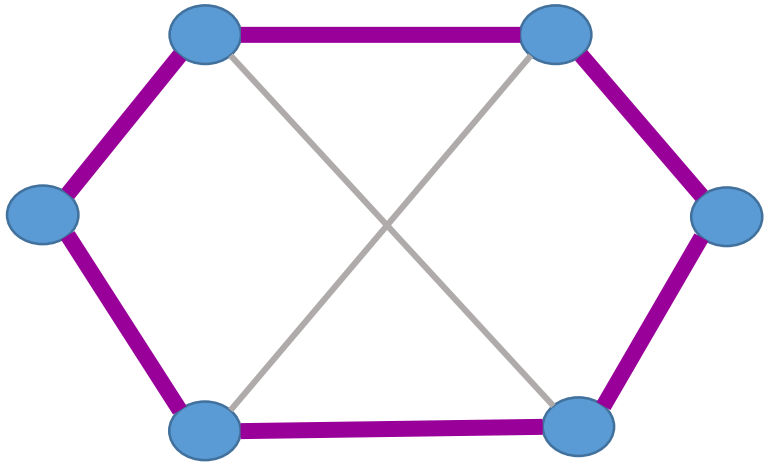
Goal: find a minimum weight **connected** subgraph



Cannot survive link failures!

Minimum 2-edge-connected spanning subgraph

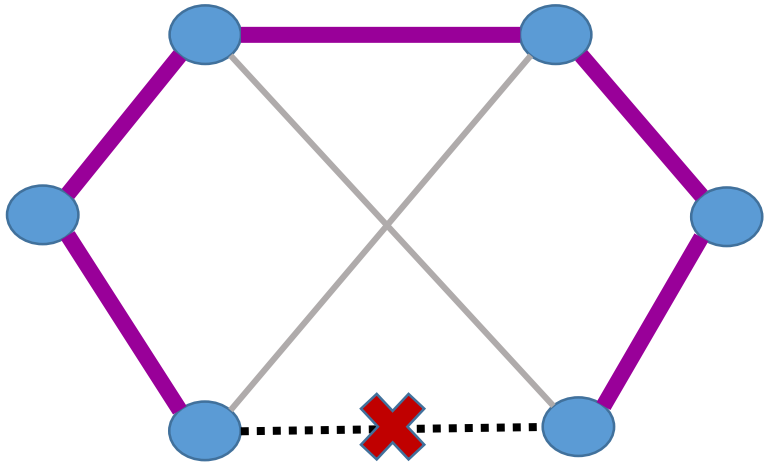
Goal: find a minimum weight **2-edge-connected** subgraph



resistant to any *single* edge failure

Minimum 2-edge-connected spanning subgraph

Goal: find a minimum weight **2-edge-connected** subgraph



resistant to any *single* edge failure

Minimum **2-edge-connected** spanning subgraph

Our goal: find the minimum weight **2-edge-connected** spanning subgraph (*2-ECSS*)

- Central problem in **network design**
- Well-studied in the **sequential** setting: 2-approximations [Khuller and Vishkin 1994, Jain 2001]

Minimum 2-edge-connected spanning subgraph

Our goal: find the minimum weight **2-edge-connected** spanning subgraph (*2-ECSS*)

What about a **distributed** algorithm?

Minimum 2-edge-connected spanning subgraph

Our goal: find the minimum weight **2-edge-connected** spanning subgraph (*2-ECSS*)

What about a **distributed** algorithm?

CONGEST model:

n vertices, **synchronous** rounds, $\Theta(\log n)$-bit messages

Previous work & our results

Previous work

Round complexity	Approximation	Notes	Reference
$O(n)$	3	deterministic	Censor-Hillel and Dory, 17
$\tilde{\Omega}(D+\sqrt{n})$	Any polynomial α	randomized	Censor-Hillel and Dory, 17
$\tilde{O}(D+\sqrt{n})$	$O(\log n)$	randomized	Dory, 18

D = diameter

Previous work

Round complexity	Approximation	Notes	Reference
$O(n)$	3	deterministic	Censor-Hillel and Dory, 17
$\tilde{\Omega}(D+\sqrt{n})$	Any polynomial α	randomized	Censor-Hillel and Dory, 17
$\tilde{O}(D+\sqrt{n})$	$O(\log n)$	randomized	Dory, 18

D = diameter

Can we get the best of both worlds?

Previous work

Round complexity	Approximation	Notes	Reference
$O(n)$	3	deterministic	Censor-Hillel and Dory, 17
$\tilde{\Omega}(D+\sqrt{n})$	Any polynomial α	randomized	Censor-Hillel and Dory, 17
$\tilde{O}(D+\sqrt{n})$	$O(\log n)$	randomized	Dory, 18

Our Result: near-optimal algorithm

$\tilde{O}(D+\sqrt{n})$	$9+\epsilon$	deterministic	

*We recently improved the approximation to $5+\epsilon$

Our First Result: near-optimal algorithm

Round complexity	Approximation	Notes	
$\tilde{O}(D+\sqrt{n})$	$9+\epsilon$	deterministic	

Our Second Result: beyond worst-case graphs

Round complexity	Approximation	Graph Family
$\tilde{O}(D)$	$O(\log n)$	planar, bounded genus, bounded path-width, bounded tree-width
$\tilde{O}(D^2)$	$O(\log n)$	excluded minor
$2^{O(\sqrt{\log n})}$	$O(\log n)$	Erdos-Renyi random graphs

Our First Result: near-optimal algorithm

Round complexity	Approximation	Notes	
$\tilde{O}(D+\sqrt{n})$	$9+\epsilon$	deterministic	

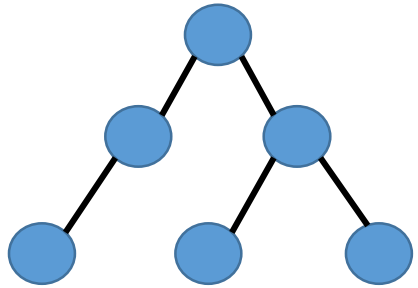
Our Second Result: beyond worst-case graphs

Round complexity	Approximation	Graph Family
$\tilde{O}(D)$	$O(\log n)$	planar, bounded genus, bounded path-width, bounded tree-width
$\tilde{O}(D^2)$	$O(\log n)$	excluded minor
$2^{O(\sqrt{\log n})}$	$O(\log n)$	Erdos-Renyi random graphs

The Algorithm

The Algorithm

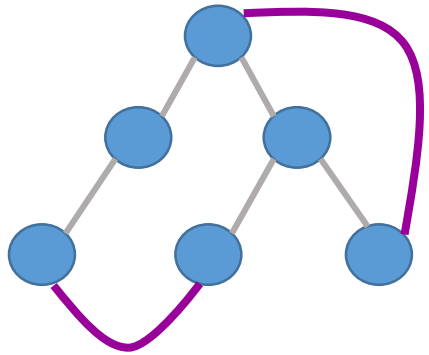
- We compute a minimum spanning tree T
- We **augment** its connectivity to 2



The Algorithm

- We compute a minimum spanning tree T
- We **augment** its connectivity to 2

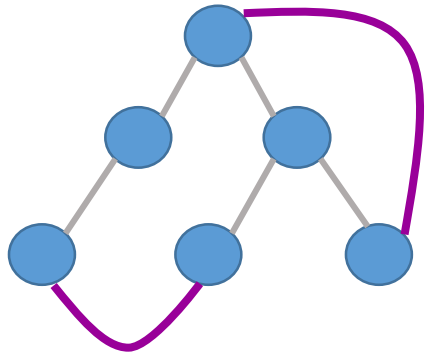
The tree augmentation problem (TAP):
Find a **minimum cost set of edges A** such that
$T \cup A$ is 2-edge-connected



The Algorithm

- We compute a minimum spanning tree T
- We **augment** its connectivity to 2

The tree augmentation problem (TAP):
Find a **minimum cost set of edges A** such that
$T \cup A$ is 2-edge-connected



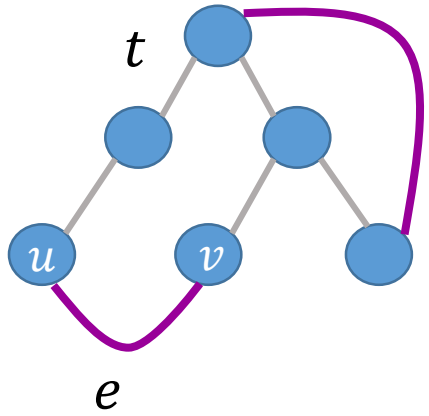
An α-approximation for **TAP** → An $(\alpha + 1)$-approximation for **2-ECSS**

TAP is a Set cover problem

The tree augmentation problem (TAP):

Find a **minimum cost set of edges A** such that **$T \cup A$ is 2-edge-connected**

- A non-tree edge e **covers** a tree edge t if $(T \setminus t) \cup e$ is connected

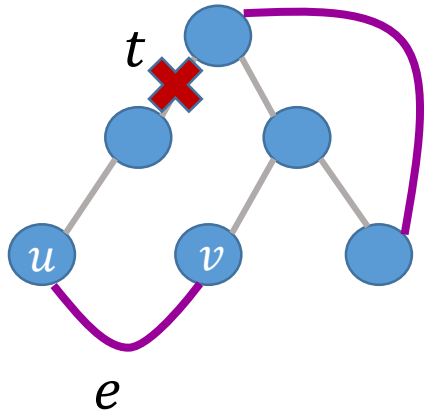


TAP is a Set cover problem

The tree augmentation problem (TAP):

Find a **minimum cost set of edges A** such that **$T \cup A$ is 2-edge-connected**

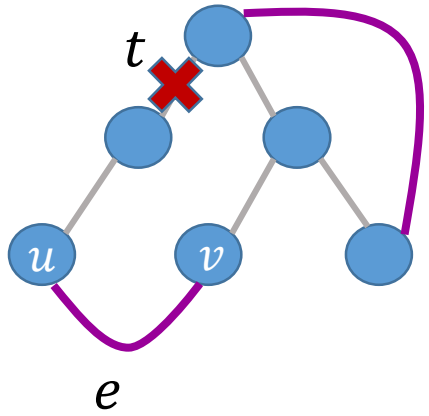
- A non-tree edge e **covers** a tree edge t if $(T \setminus t) \cup e$ is connected
- The edge $e = \{u, v\}$ **covers** all the tree edges in the $u - v$ path in T



TAP is a Set cover problem

The tree augmentation problem (TAP):
Find a **minimum cost set of edges A** such that
$T \cup A$ is 2-edge-connected

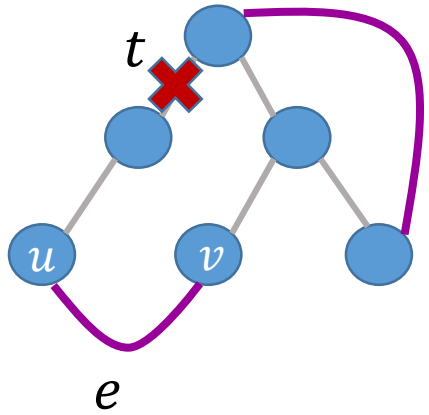
- A non-tree edge e **covers** a tree edge t if $(T \setminus t) \cup e$ is connected
- The edge $e = \{u, v\}$ **covers** all the tree edges in the $u - v$ path in T
- The goal: **cover all the tree edges** with **minimum cost** set of non-tree edges

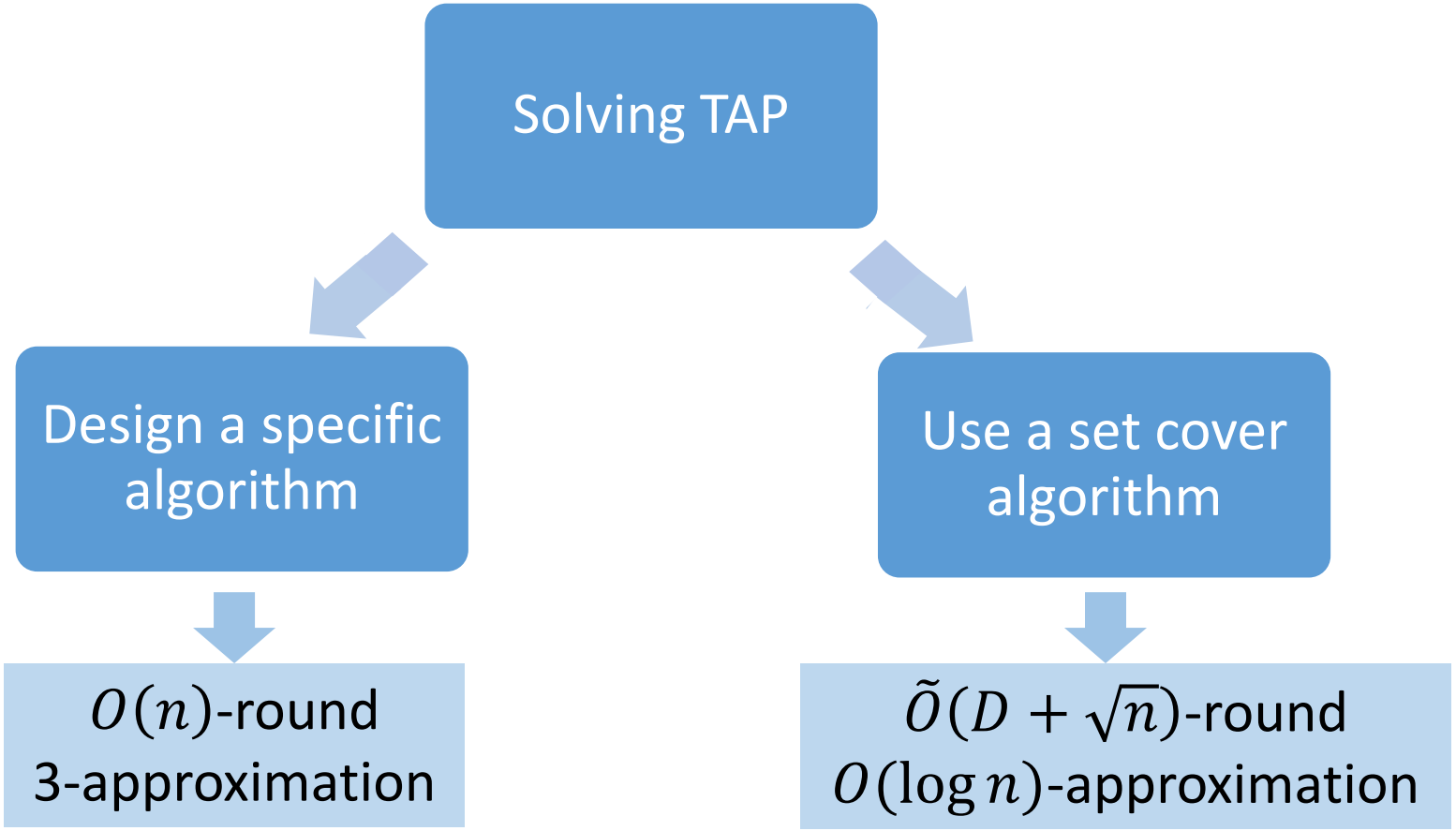


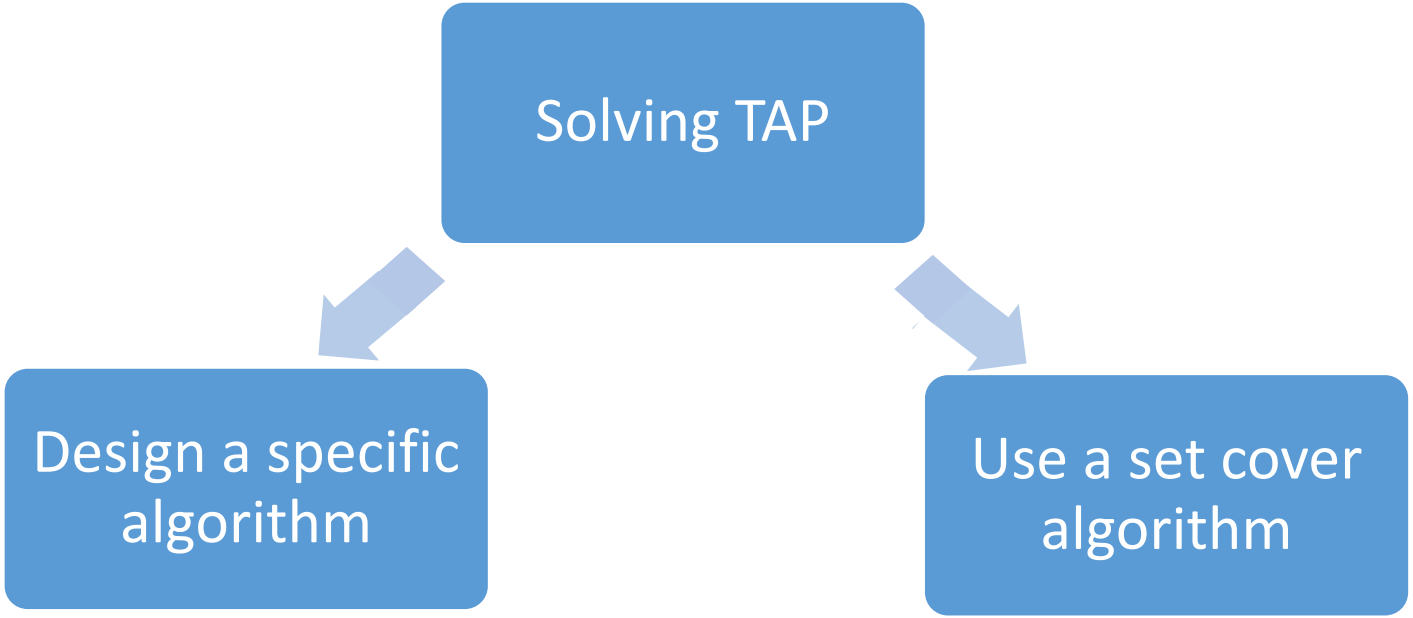
TAP is a Set cover problem

The tree augmentation problem (TAP):
Find a **minimum cost set of edges A** such that
$T \cup A$ is 2-edge-connected

- The goal: **cover all the tree edges** with minimum cost set of **non-tree edges**
- Special case of set cover: cover a **universe** with minimum cost collection of **sets**




Solving TAP
Design a specific algorithm
Use a set cover algorithm
$O(n)$-round
3-approximation
$\tilde{O}(D+\sqrt{n})$-round
$O(\log n)$-approximation



Our approach:

Use a **set cover** algorithm that exploits the **specific structure** of TAP

$\tilde{O}(D + \sqrt{n})$-round $(9 + \epsilon)$-approximation

Exploiting the set cover structure

- Set cover with **small neighbourhood covers (SNC)**
- The SNC algorithm

Exploiting the set cover structure

- Set cover with **small neighbourhood covers (SNC)**
- The SNC algorithm

Set cover with **small neighbourhood covers** (SNC)

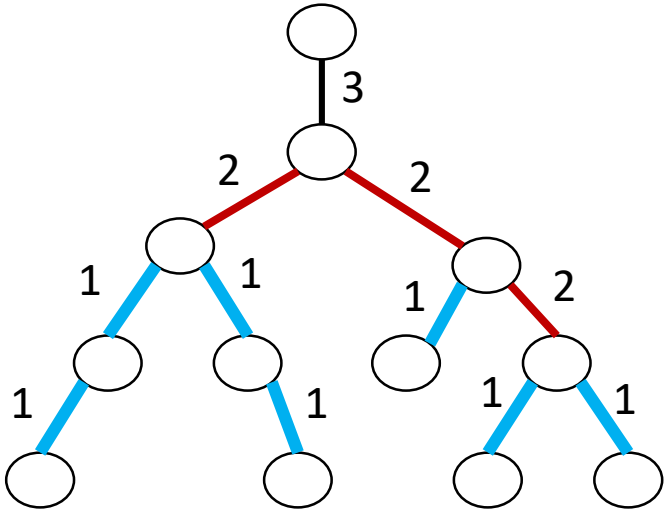
- The **SNC property** was introduced in

[Agarwal, Chakaravarthy, Choudhury, Roy, Sabharwal, 2018]

- Sequential and parallel algorithms that exploit this property
- Examples: vertex cover, interval cover, tree cover, …

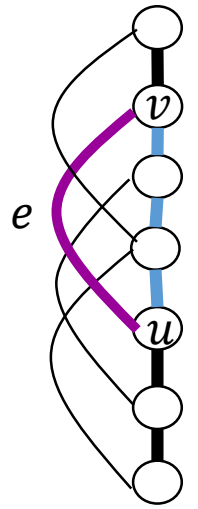
Simplifying the problem

- It's "enough" to look at **paths**.
- We decompose the graph into $O(\log n)$ **layers** of disjoint paths.



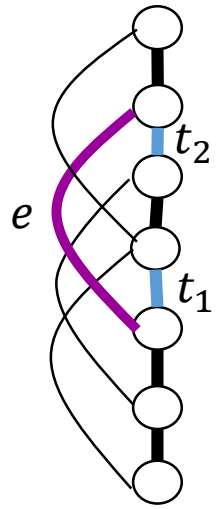
Set cover with **small neighbourhood covers** (SNC)
[Agarwal et al., 2018]

- The non-tree edge $e = \{u, v\}$ **covers** all the tree edges in the $u - v$ path in T.



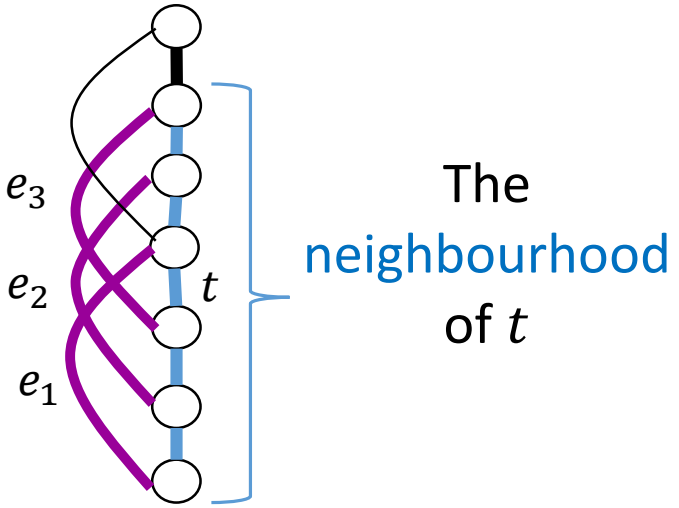
Set cover with **small neighbourhood covers** (SNC) [Agarwal et al., 2018]

- The non-tree edge $e = \{u, v\}$ **covers** all the tree edges in the $u - v$ path in T.
- The tree edges t_1, t_2 are **neighbors** if there is a non-tree edge e that covers both of them.



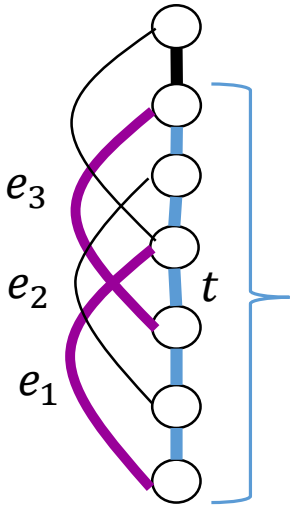
Set cover with **small neighbourhood covers** (SNC)
[Agarwal et al., 2018]

- The tree edges t_1, t_2 are **neighbors** if there is a non-tree edge e that covers both of them.
- The **neighbourhood** of t = all its neighbors



Set cover with **small neighbourhood covers** (SNC)
[Agarwal et al., 2018]

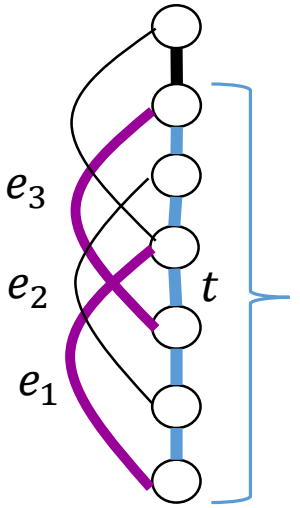
- The tree edges t_1, t_2 are **neighbors** if there is a non-tree edge e that covers both of them.
- The **neighbourhood** of t = all its neighbors
- **The SNC property** = the neighbourhood of t can be covered by **2** non-tree edges



The neighbourhood of t

Set cover with **small neighbourhood covers** (SNC) [Agarwal et al., 2018]

- The tree edges t_1, t_2 are **neighbors** if there is a non-tree edge e that covers both of them.
- The **neighbourhood** of t = all its neighbors
- **The SNC property** = the neighbourhood of t can be covered by **2** non-tree edges
- These edges are called the **petals** of t.



The neighbourhood of t

Exploiting the set cover structure

- Set cover with **small neighbourhood covers (SNC)**
- The SNC algorithm

Exploiting the set cover structure

- Set cover with **small neighbourhood covers (SNC)**
- The SNC algorithm

The SNC algorithm [Agarwal et al., 2018]

Forward phase:

Choose a set of "good" edges $\boldsymbol{A}$ that **covers all tree edges**

Reverse-delete phase:

Choose a cover $\boldsymbol{B} \subseteq A$ such that all tree edges are **covered at most 4 times** by B

The SNC algorithm [Agarwal et al., 2018]

Forward phase:
Choose a set of "good" edges A that **covers all tree edges**

Reverse-delete phase:
Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Gives a $(4+\epsilon)$-**approximation** for TAP in a path

Translates to a $(4+\epsilon)2+1 = (9+\epsilon)$-**approximation** for 2-ECSS

The SNC algorithm [Agarwal et al., 2018]

Forward phase:
Choose a set of "good" edges A that **covers all tree edges**

Reverse-delete phase:
Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

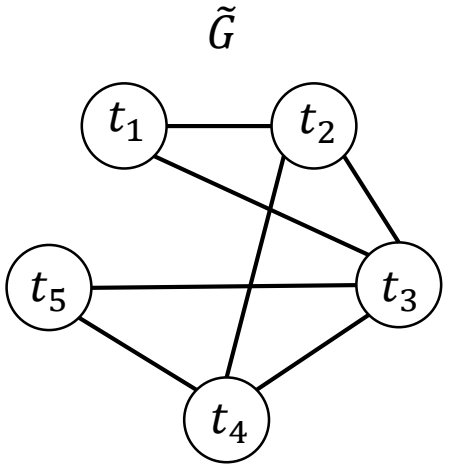
Gives a $(4+\epsilon)$-**approximation** for TAP in a path

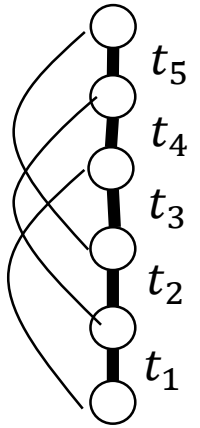
Translates to a $(4+\epsilon)2+1 = (9+\epsilon)$-**approximation** for 2-ECSS

The reverse-delete phase

Choose a cover $\boldsymbol{B} \subseteq A$ such that all tree edges are **covered at most 4 times** by B

- Tree edges t_1, t_2 are **neighbours** if there is a non-tree edge that covers both of them
- Defines a **neighbourhood graph** $\tilde{G}$

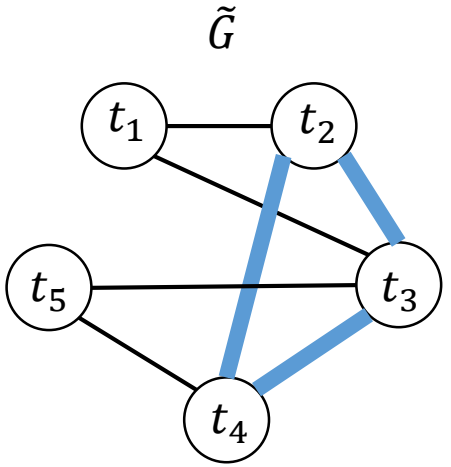


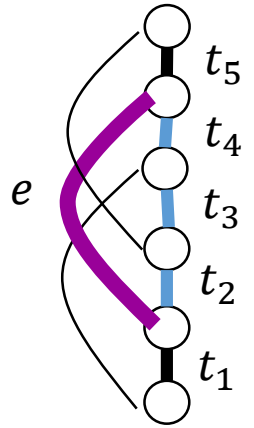


The reverse-delete phase

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

- Tree edges t_1, t_2 are **neighbours** if there is a non-tree edge that covers both of them
- Defines a **neighbourhood graph** $\tilde{G}$

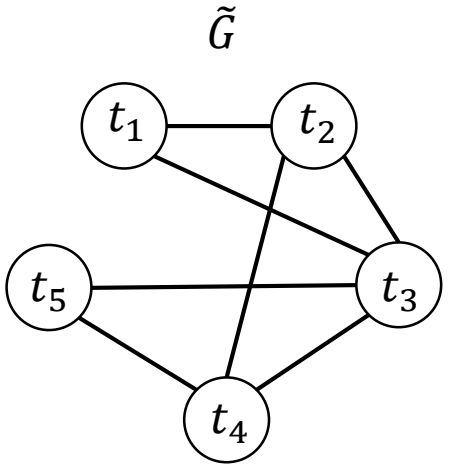


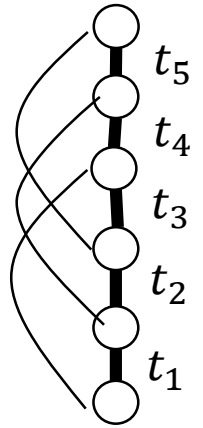


The reverse-delete phase

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

- Tree edges t_1, t_2 are **neighbours** if there is a non-tree edge that covers both of them
- Defines a **neighbourhood graph** $\tilde{G}$
- A **maximal independent set** (MIS) in $\tilde{G}$: a maximal set of non-neighboring tree edges

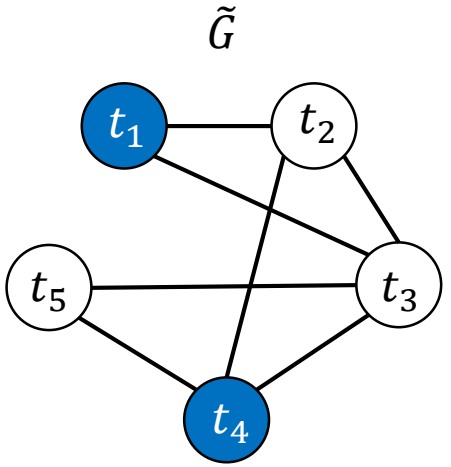


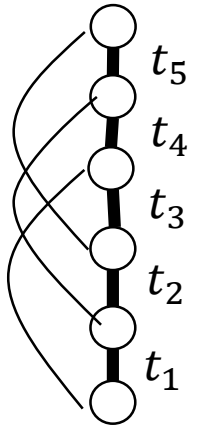


The reverse-delete phase

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

- Tree edges t_1, t_2 are **neighbours** if there is a non-tree edge that covers both of them
- Defines a **neighbourhood graph** $\tilde{G}$
- A **maximal independent set** (MIS) in $\tilde{G}$: a maximal set of non-neighboring tree edges



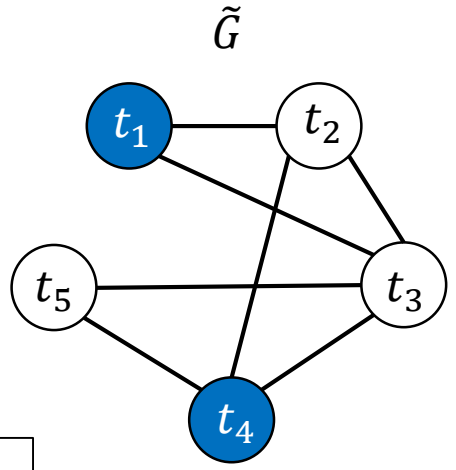


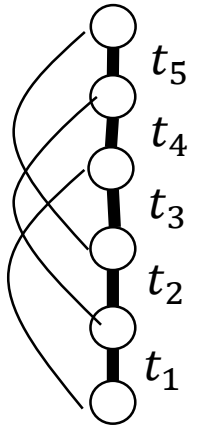
The reverse-delete phase

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

The algorithm:

- Find a **maximal independent set** M in $\tilde{G}$
- For each $t \in M$, add its 2 **petals** to B





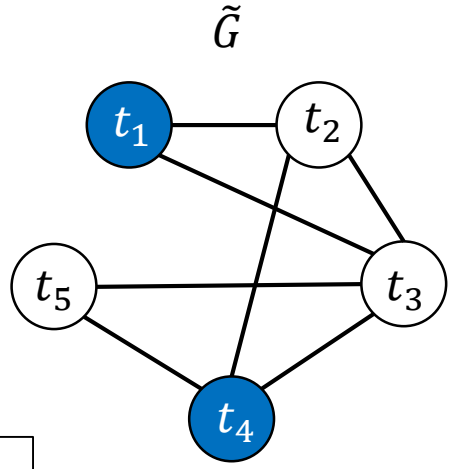
The **petals** of a tree edge t =
at most 2 edges that cover the neighborhood of t

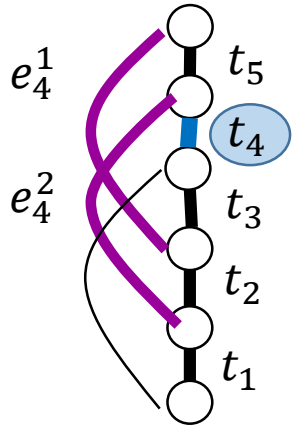
The reverse-delete phase

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

The algorithm:

- Find a **maximal independent set** M in $\tilde{G}$
- For each $t \in M$, add its 2 **petals** to B





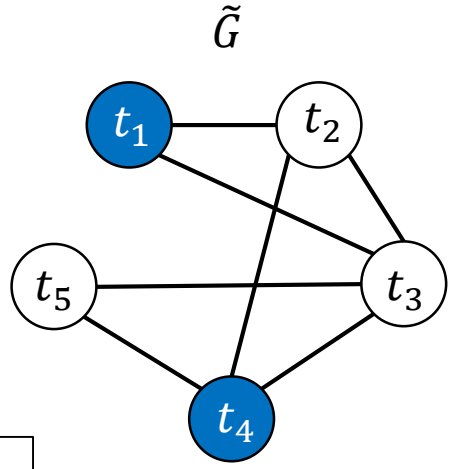
The **petals** of a tree edge t =
at most 2 edges that cover the neighborhood of t

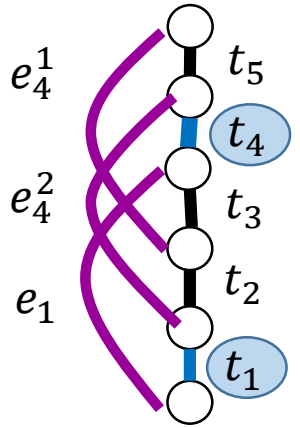
The reverse-delete phase

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

The algorithm:

- Find a **maximal independent set** M in $\tilde{G}$
- For each $t \in M$, add its 2 **petals** to B





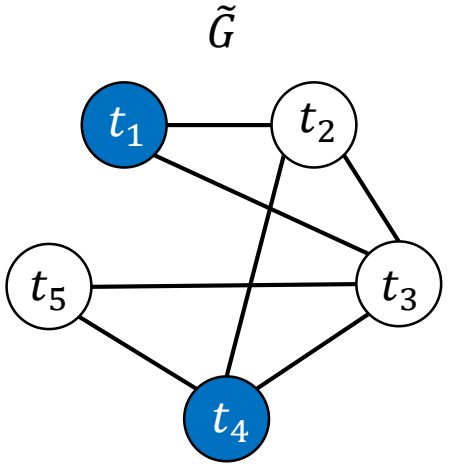
The **petals** of a tree edge t =
at most 2 edges that cover the neighborhood of t

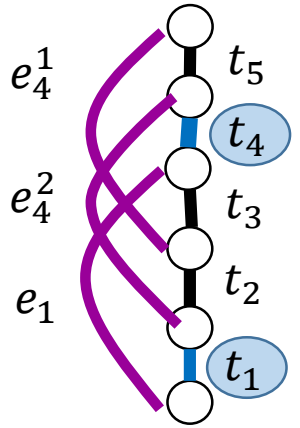
The reverse-delete phase: correctness

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

The algorithm:

- Find a **maximal independent set** M in $\tilde{G}$
- For each $t \in M$, add its 2 **petals** to B





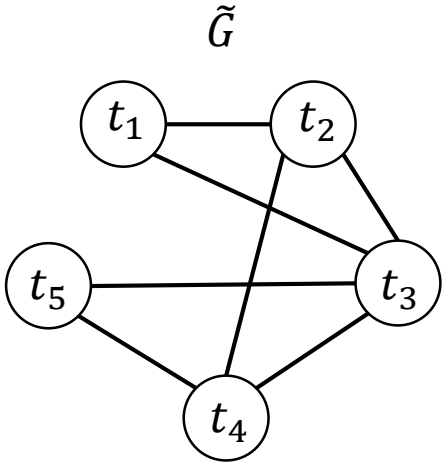
B is a cover:

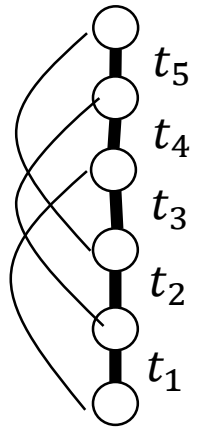
- Each tree edge t has a neighbor t' in M
- The petals of t' cover t

The reverse-delete phase: correctness

Choose a cover $\boldsymbol{B} \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Claim: each tree edge t has at most 2 neighbors in M



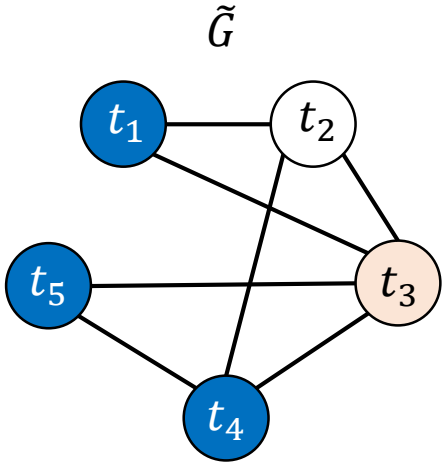


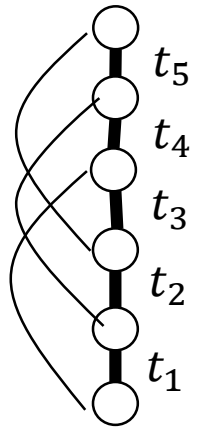
The reverse-delete phase: correctness

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Claim: each tree edge t has at most 2 neighbors in M

- Assume that t_3 has **3 neighbors** in M



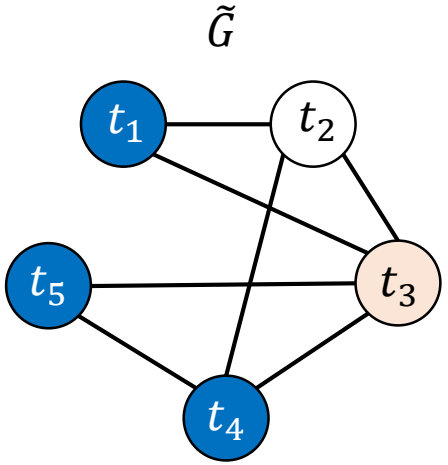


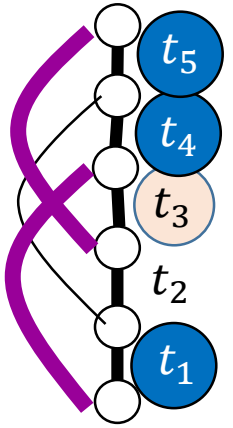
The reverse-delete phase: correctness

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Claim: each tree edge t has at most 2 neighbors in M

- Assume that t_3 has **3 neighbors** in M
- The **2 petals** of t_3 cover all its neighbors





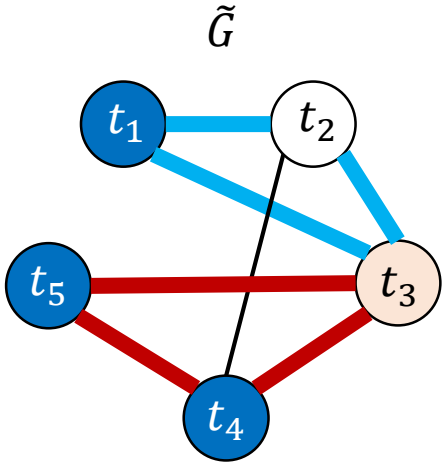
The reverse-delete phase: correctness

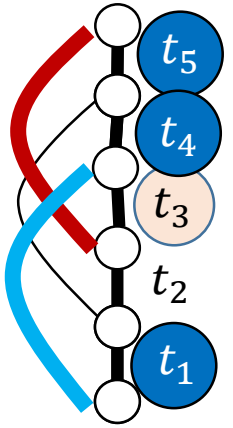
Choose a cover $\boldsymbol{B} \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Claim: each tree edge t has at most 2 neighbors in M

- Assume that t_3 has **3 neighbors** in M
- The **2 petals** of t_3 cover all its neighbors

- At least **2 edges in M** are covered by the same petal → **neighbors**
- Contradiction to independence of M

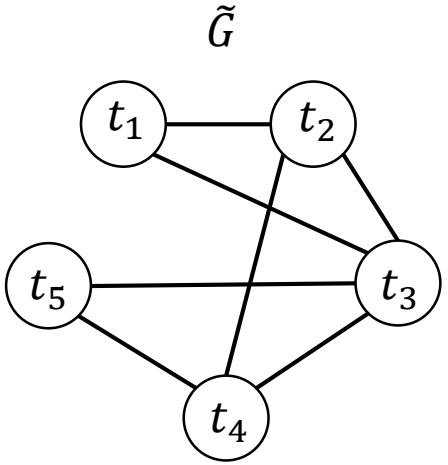


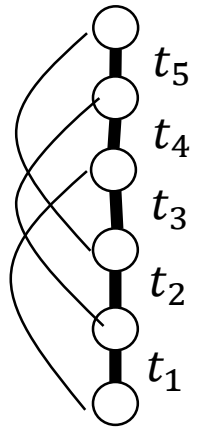


The reverse-delete phase: correctness

Choose a cover $\boldsymbol{B} \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Claim: each tree edge t has at most 2 neighbors in M



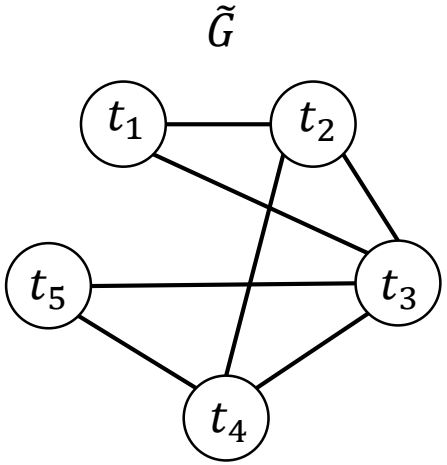


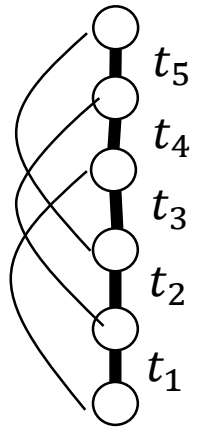
The reverse-delete phase: correctness

Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Claim: each tree edge t has at most 2 neighbors in M

- Each of t's neighbors in M adds its **2 petals** to B
- t is covered at most **4 times**





The SNC algorithm [Agarwal et al., 2018]

Forward phase:
Choose a set of "good" edges A that **covers all tree edges**

Reverse-delete phase:
Choose a cover $B \subseteq A$ such that all tree edges are **covered at most 4 times** by B

Gives a $(4+\epsilon)$-**approximation** for TAP in a path

Translates to a $(4+\epsilon)2+1 = (9+\epsilon)$-**approximation** for 2-ECSS

Distributed implementation

Toolkit:

- LCA labels
- Decomposition
- Layering
- …

Main challenges:

- The **set cover** graph is not given
- Computing an MIS in the **neighborhood graph** $\tilde{G}$
- …

Our First Result: near-optimal algorithm

Round complexity	Approximation	Notes	
$\tilde{O}(D+\sqrt{n})$	$9+\epsilon$	deterministic	

Our Second Result: beyond worst-case graphs

Round complexity	Approximation	Graph Family
$\tilde{O}(D)$	$O(\log n)$	planar, bounded genus, bounded path-width, bounded tree-width
$\tilde{O}(D^2)$	$O(\log n)$	excluded minor
$2^{O(\sqrt{\log n})}$	$O(\log n)$	Erdos-Renyi random graphs

Open questions

- Minimum **k-edge-connected** subgraph for $k > 2$
- Exploiting the **specific structure** of set cover problems