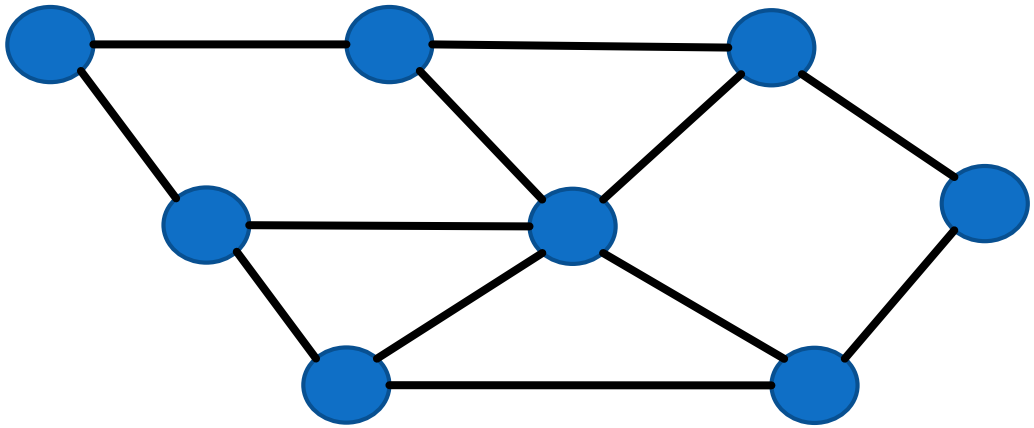
Distributed Approximation for Tree Augmentation

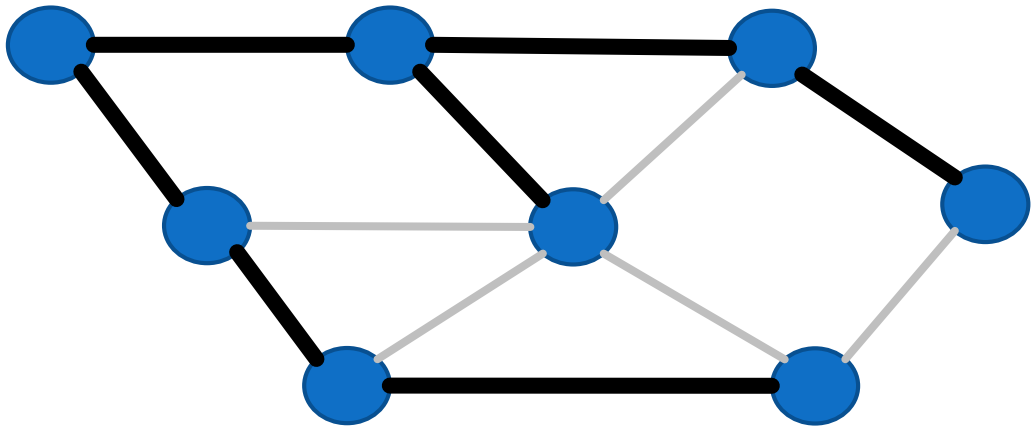
Michal Dory, Technion

Joint work with: Keren Censor-Hillel, Technion

Consider a Communication Network

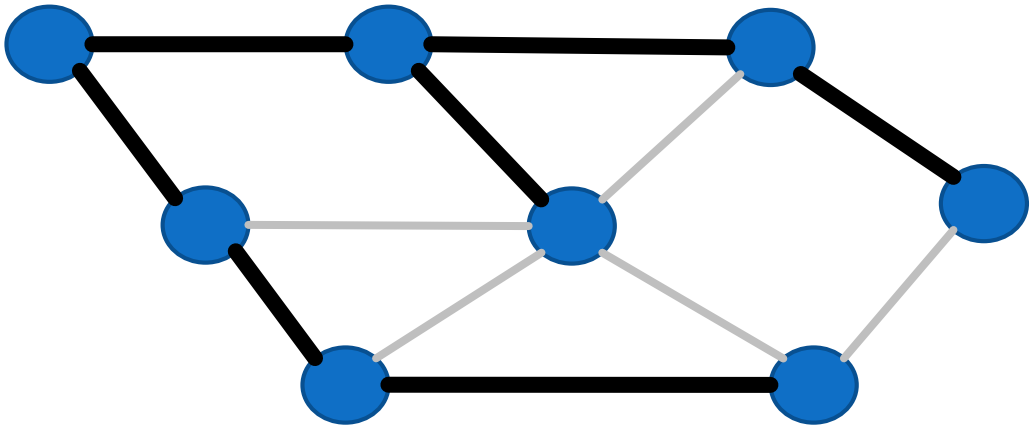


We can communicate over a spanning tree

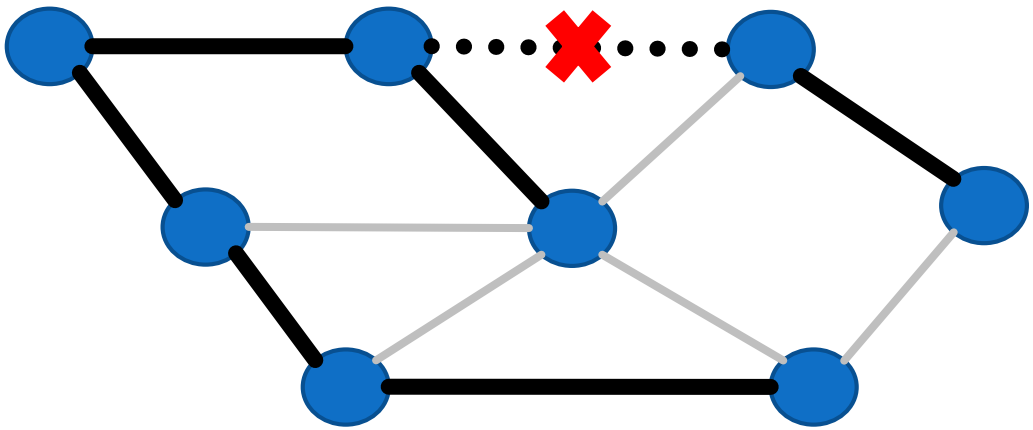


There are many constructions of minimum spanning trees (MST)

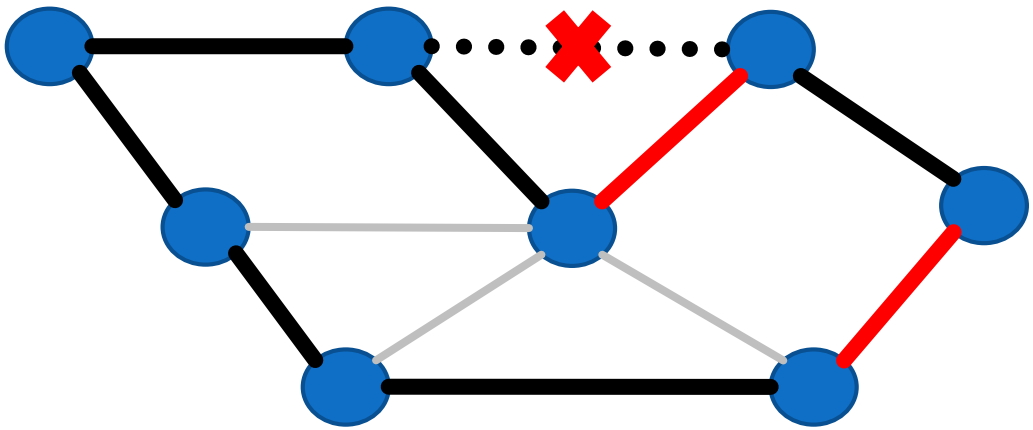
[Gallager, Humblet and Spira 83, Kutten and Peleg 95, Garay, Kutten and Peleg 98, Pandurangan, Robinson and Scquizzato 17, Elkin 17,...]



Trees cannot survive a link failure

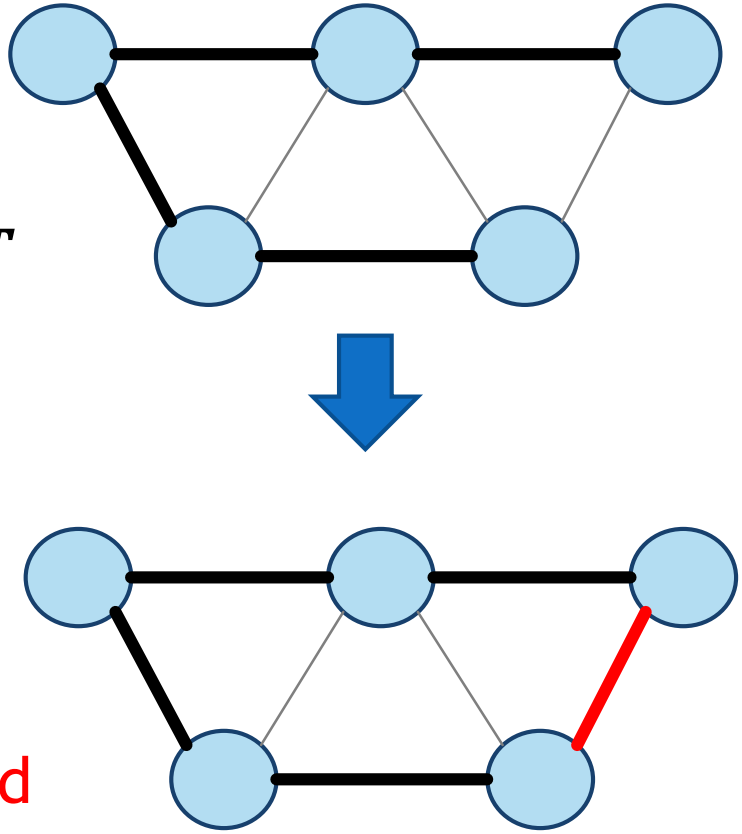


Trees cannot survive a link failure



The Tree Augmentation Problem (TAP)

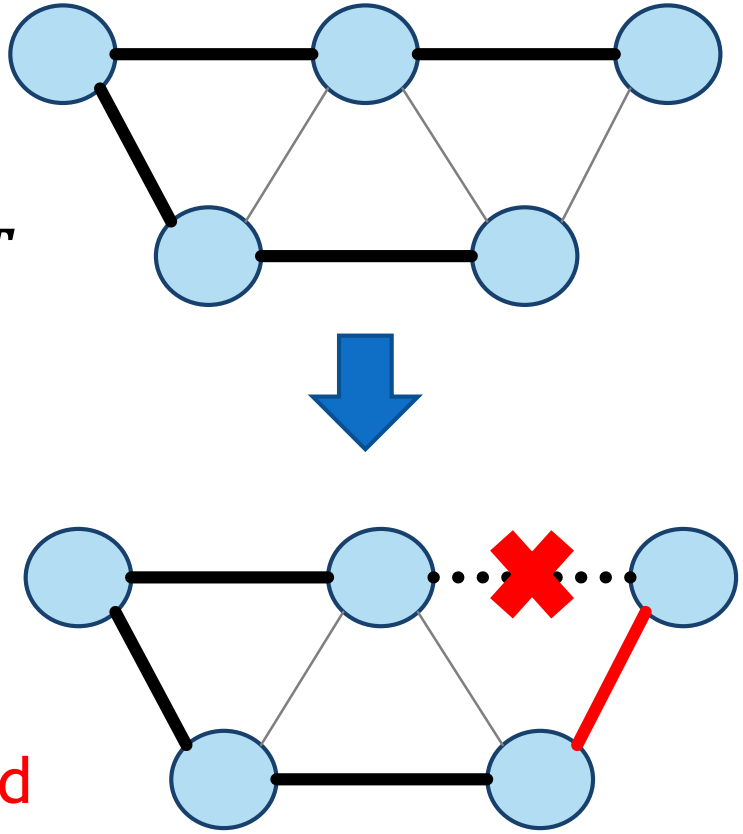
Input:
a graph G
a spanning tree $\boldsymbol{T}$



Goal:
augment $\boldsymbol{T}$ to be
2-edge-connected

The Tree Augmentation Problem (TAP)

Input:
a graph G
a spanning tree $\boldsymbol{T}$



Goal:
augment $\boldsymbol{T}$ to be
2-edge-connected

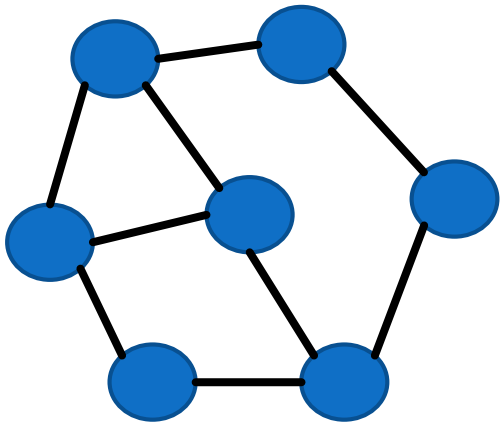
The Tree Augmentation Problem (TAP)

- A central problem in network design
- Has many **sequential** algorithms:

  [Frederickson and JáJá 81, Khuller and Thurimella 93, Goemans et al. 94, Jain 01, Kortsarz and Nutov 16, Adjiashvili 17…].
- Goal: solve TAP in the **distributed** setting.

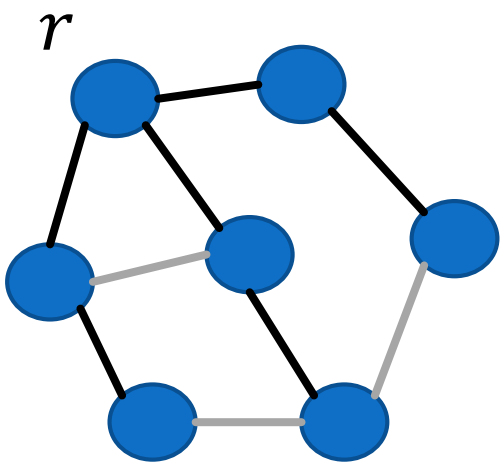
The CONGEST model

- Communication network with n processors
- Synchronous rounds
- Messages of $O(\log n)$ bits
- Time = number of rounds
- The input and output are local



The CONGEST model

- Communication network with n processors
- Synchronous rounds
- Messages of $O(\log n)$ bits
- Time = number of rounds
- The input and output are local

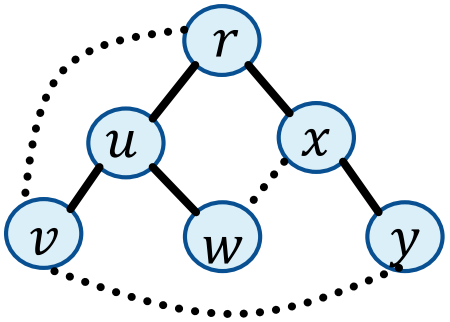


Sequential Approximation for TAP
[Khuller and Thurimella 93]

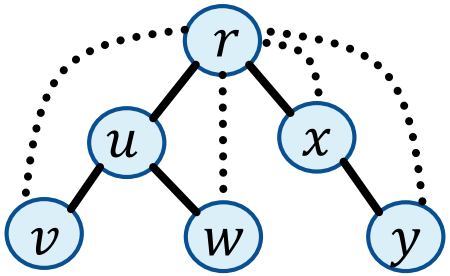
- Builds a new **virtual graph** G'
- Finds a **directed MST** in G'
- This gives a 2-approximation in G

Distributed Approximation for TAP

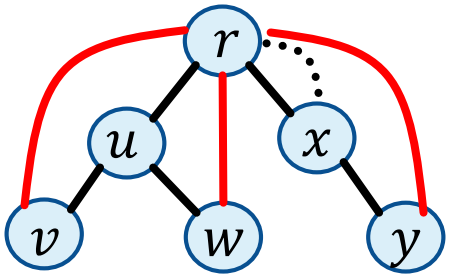
The input graph G



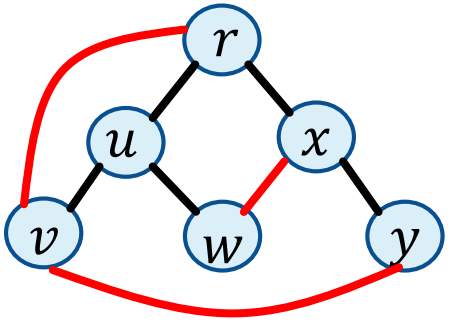
A virtual graph G'



Optimal solution in G'



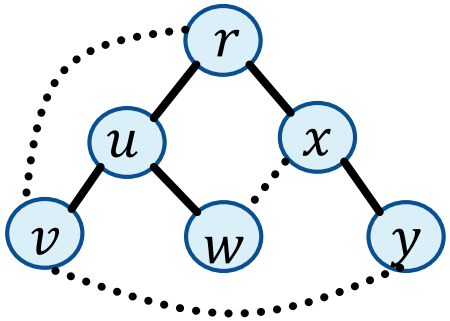
2-approximation in G



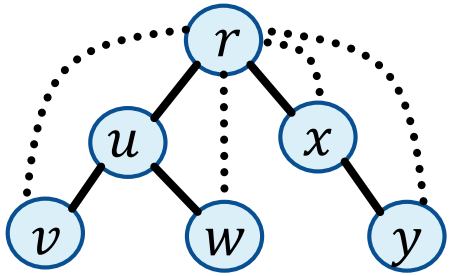
The Graph G'

- All the non-tree edges in G' are between ancestors to descendants:

The input graph G



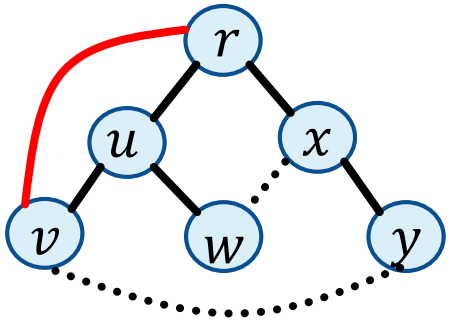
A virtual graph G'



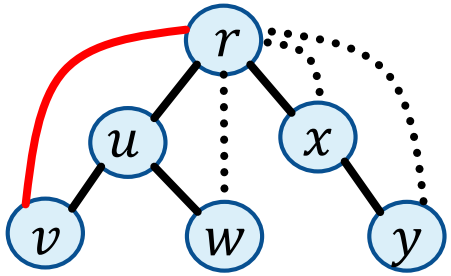
The Graph G'

- All the non-tree edges in G' are between ancestors to descendants:

The input graph G



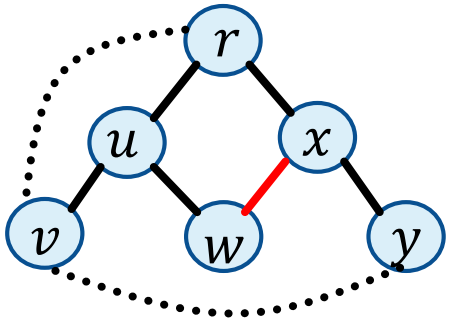
A virtual graph G'



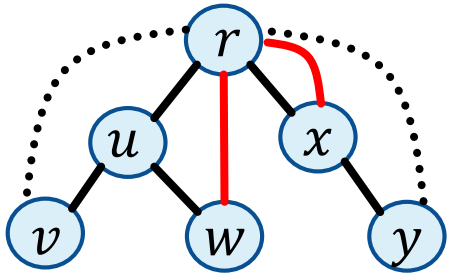
The Graph G'

- All the non-tree edges in G' are between ancestors to descendants:

The input graph G



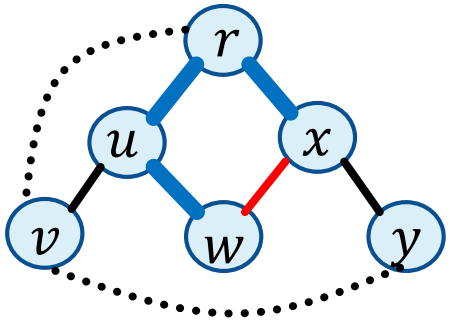
A virtual graph G'



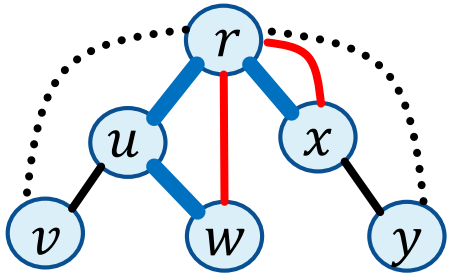
The Graph G'

- The corresponding edges in G' *cover* exactly the same tree edges:

The input graph G



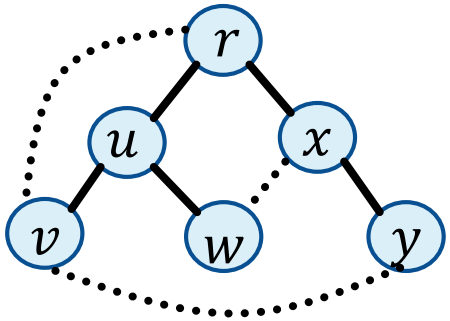
A virtual graph G'



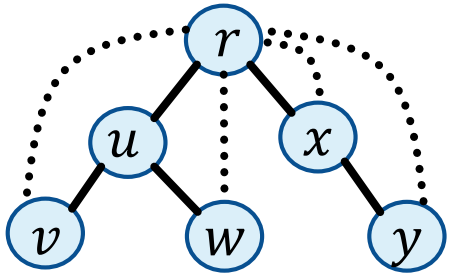
The Graph G'

- We can build G' in $O(h)$ rounds using LCA labels of $O(\log n)$ bits [Alstrup et al. 2004]

The input graph G

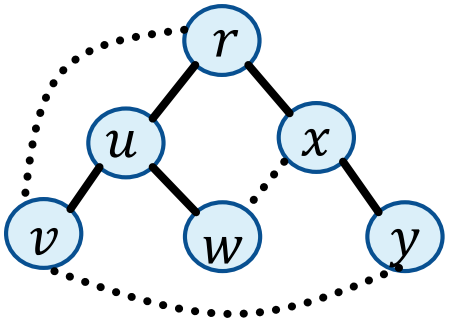


A virtual graph G'

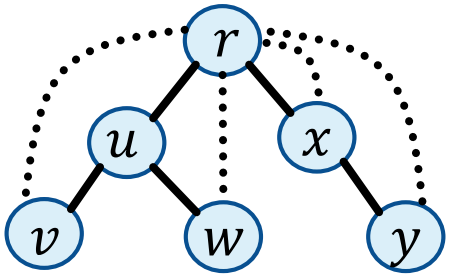


Distributed Approximation for TAP

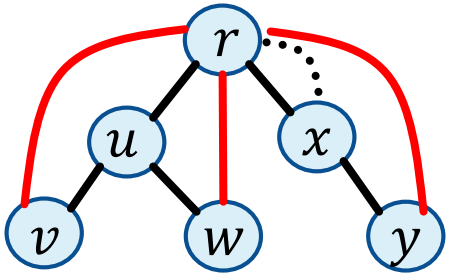
The input graph G



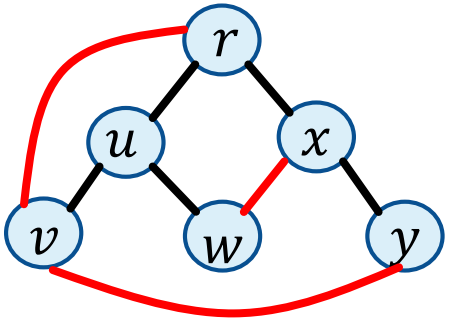
A virtual graph G'



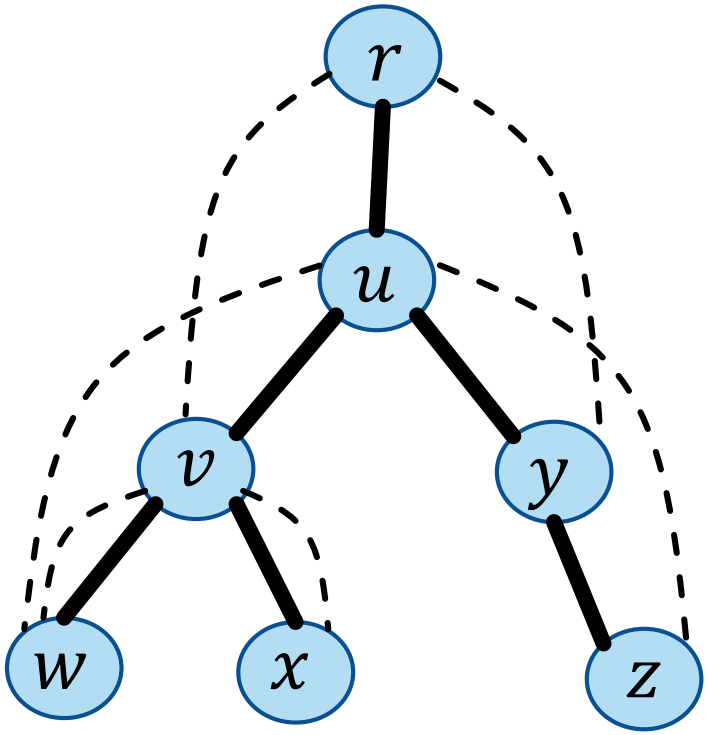
Optimal solution in G'



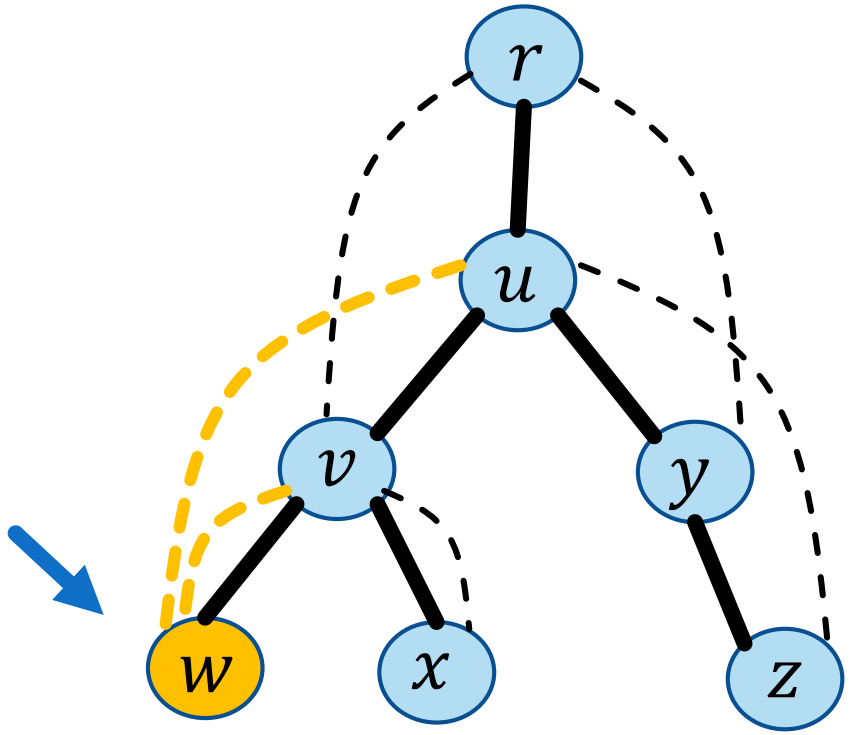
2-approximation in G



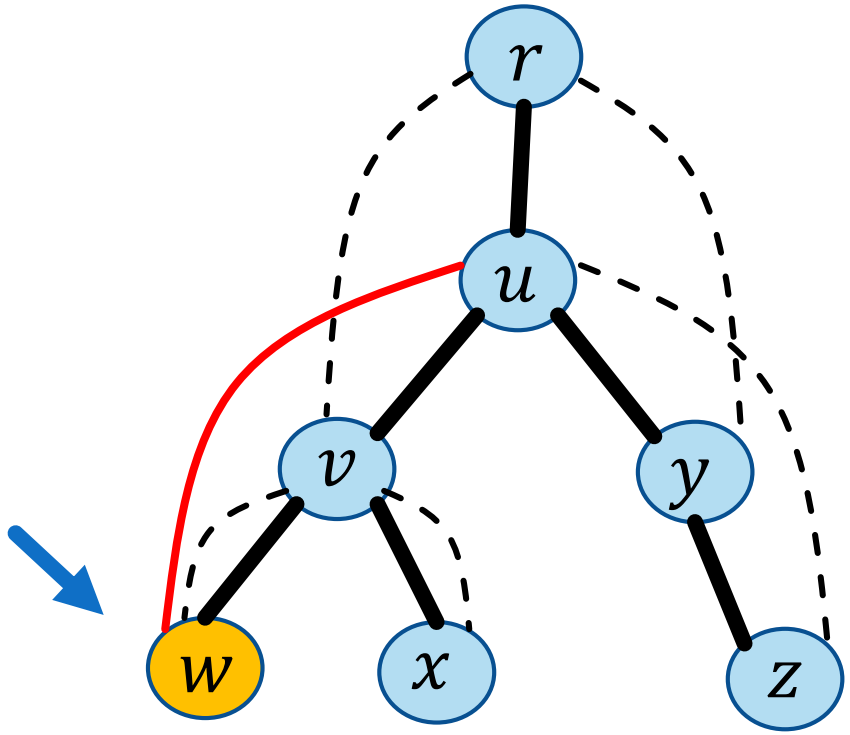
Finding an Optimal Augmentation in G'



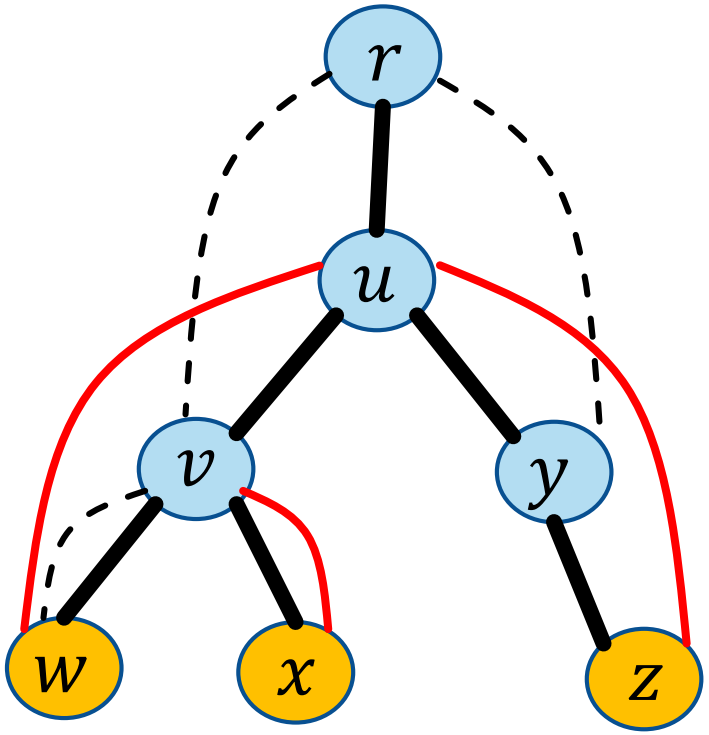
Finding an Optimal Augmentation in G'



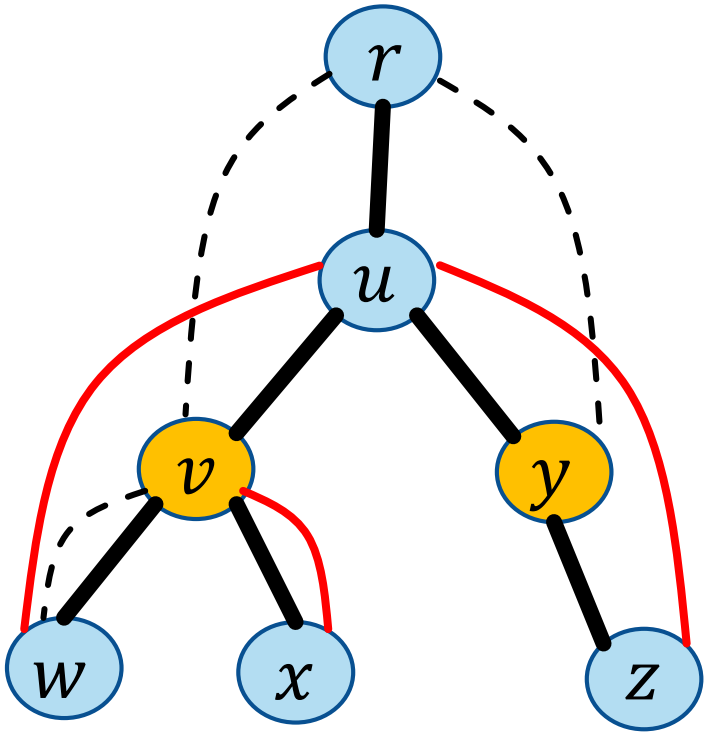
Finding an Optimal Augmentation in G'



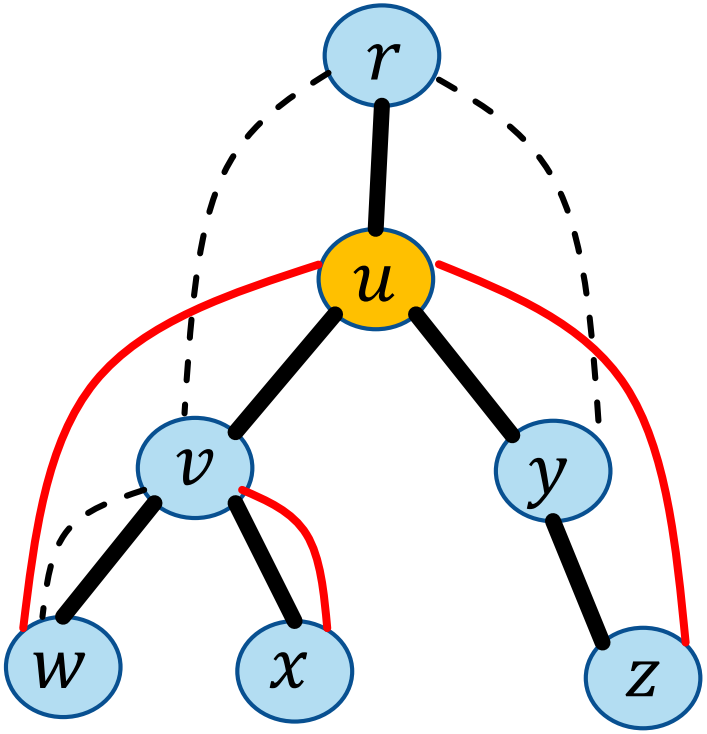
Finding an Optimal Augmentation in G'



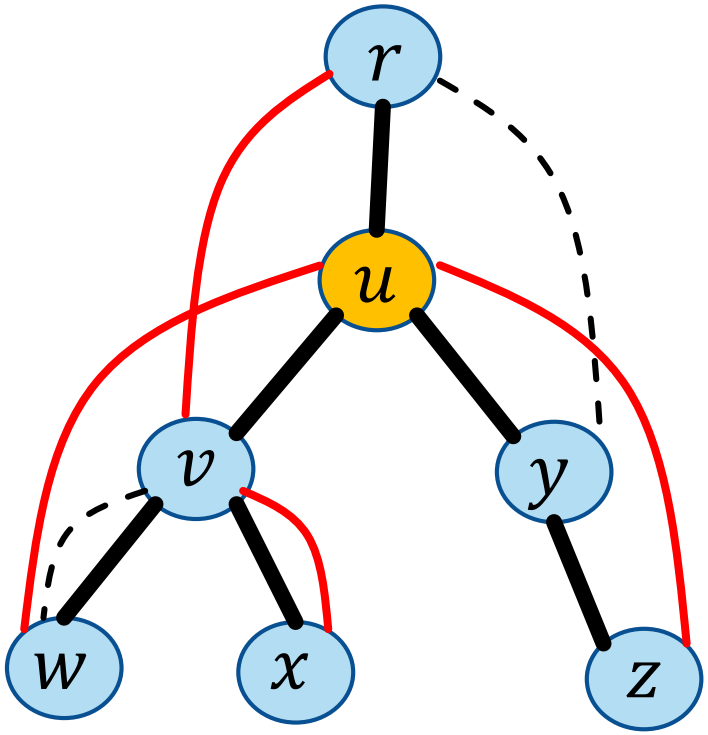
Finding an Optimal Augmentation in G'



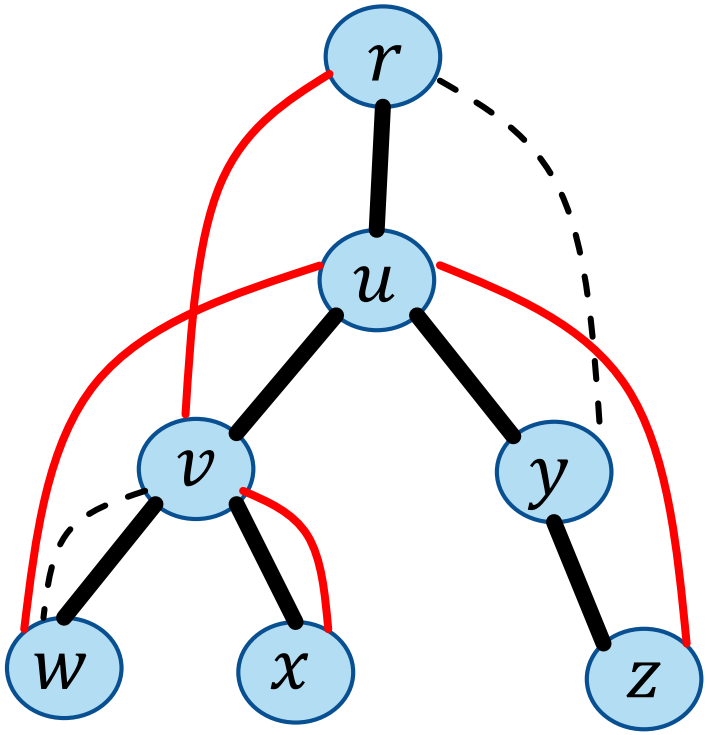
Finding an Optimal Augmentation in G'



Finding an Optimal Augmentation in G'



Finding an Optimal Augmentation in G'



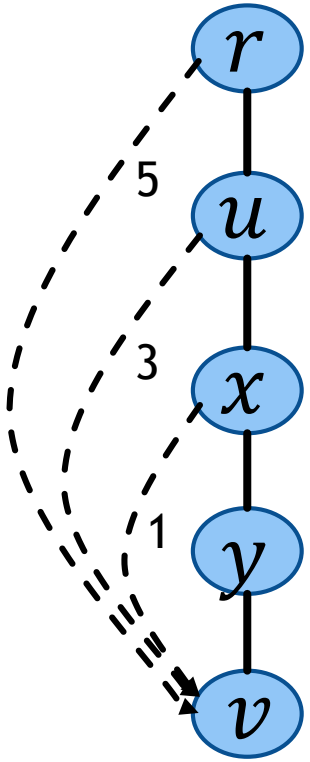
Conclusion

- There is a **2-approximation** for *unweighted* TAP in $\boldsymbol{O(h)}$ rounds, where h = height of T.
- What about the weighted case?

Weighted TAP

Problem:

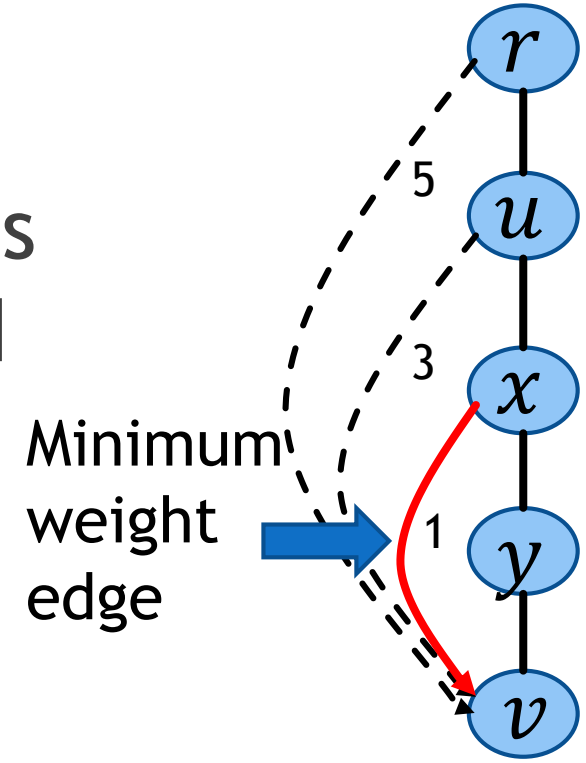
how to compare edges?



Weighted TAP

Solution:

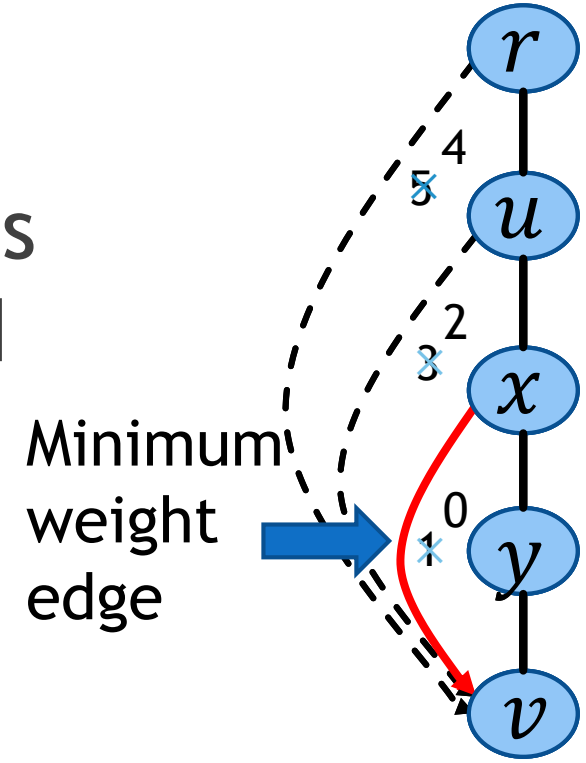
sending edges with reduced weights



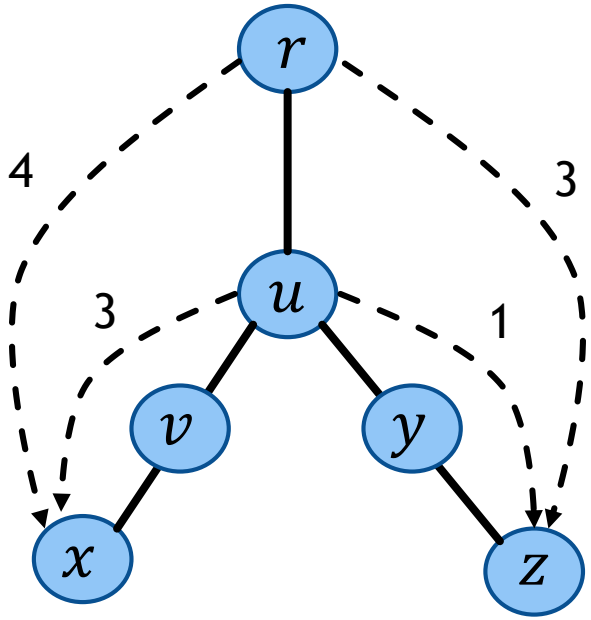
Weighted TAP

Solution:

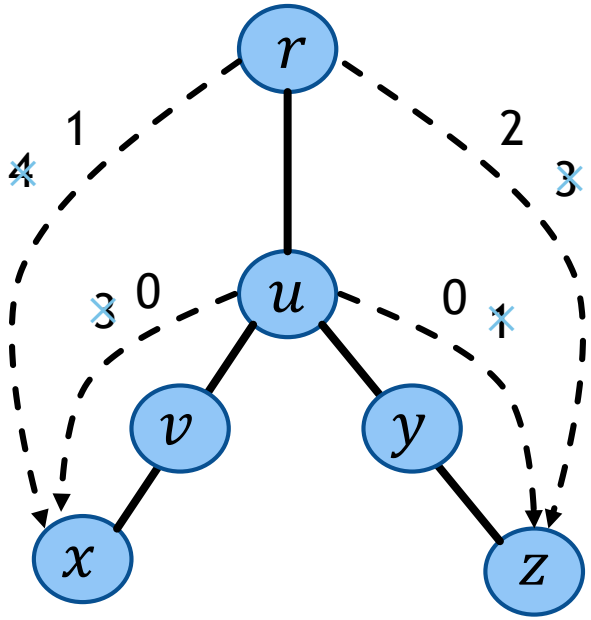
sending edges with reduced weights



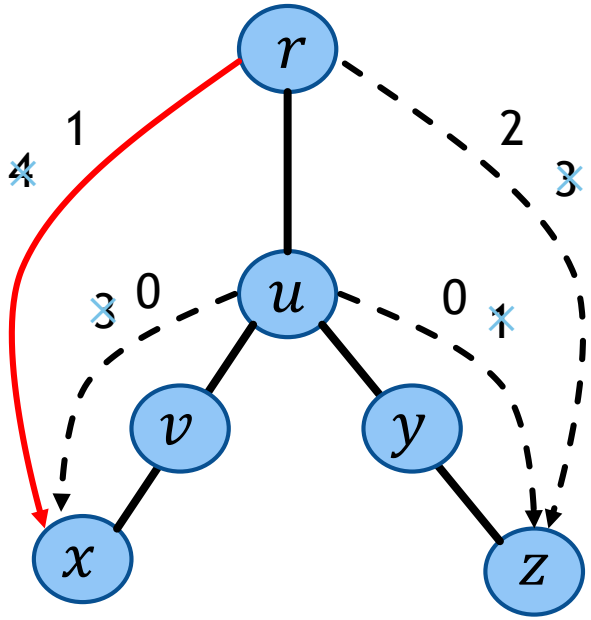
Example



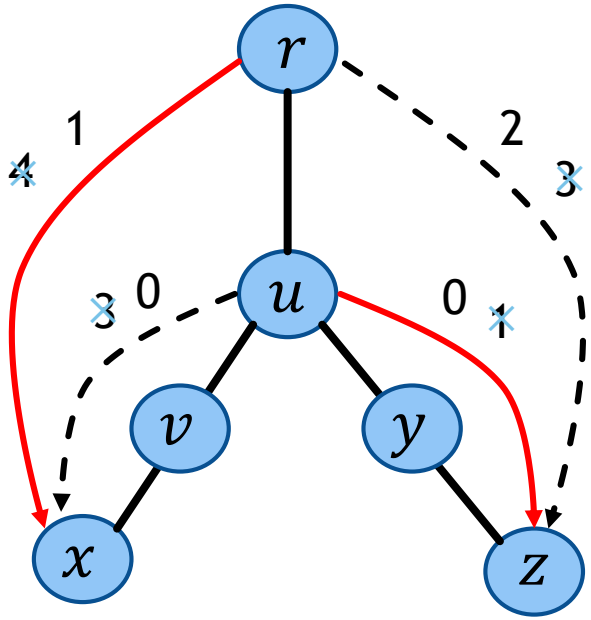
Example



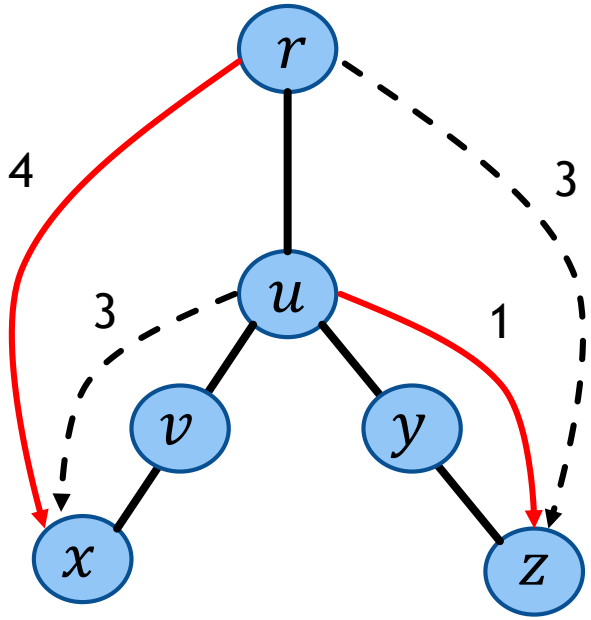
Example



Example

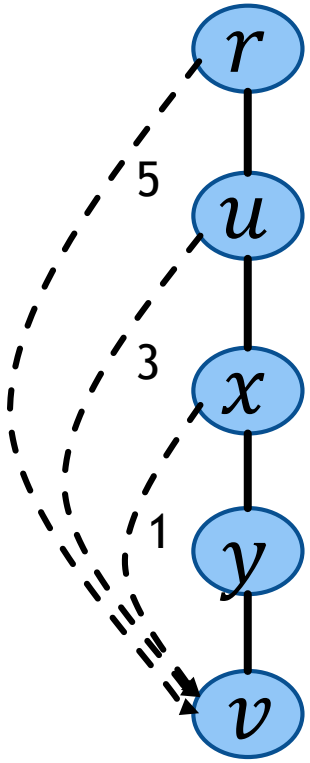


Example



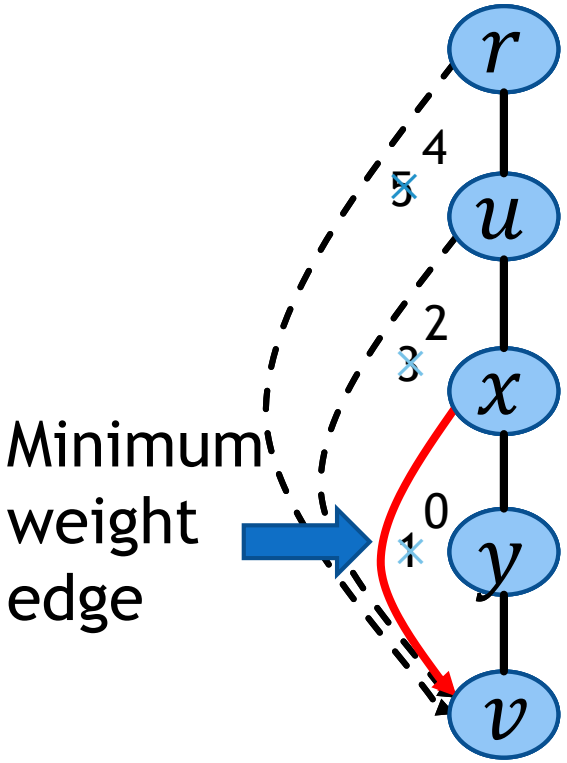
The Algorithm

Each vertex v computes the weights of the minimum weight edges that cover the path from v to each of its ancestors (at most h edges).



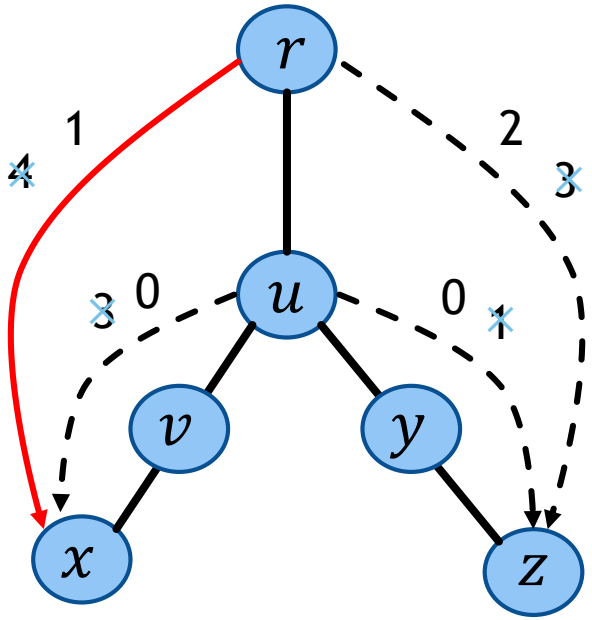
The Algorithm

It sends to its parent these edges after reducing from them the weight min_v of the minimum weight edge covering $\{v, p(v)\}$.



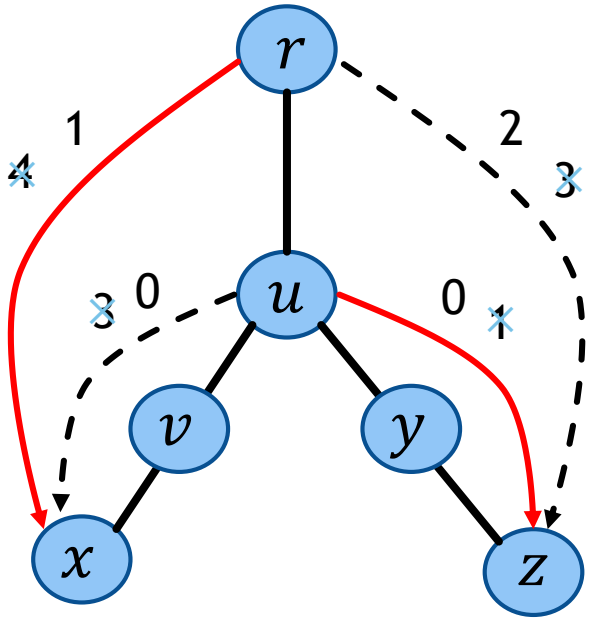
The Algorithm

At the end each child v of r adds to the augmentation the edge having weight min_v.



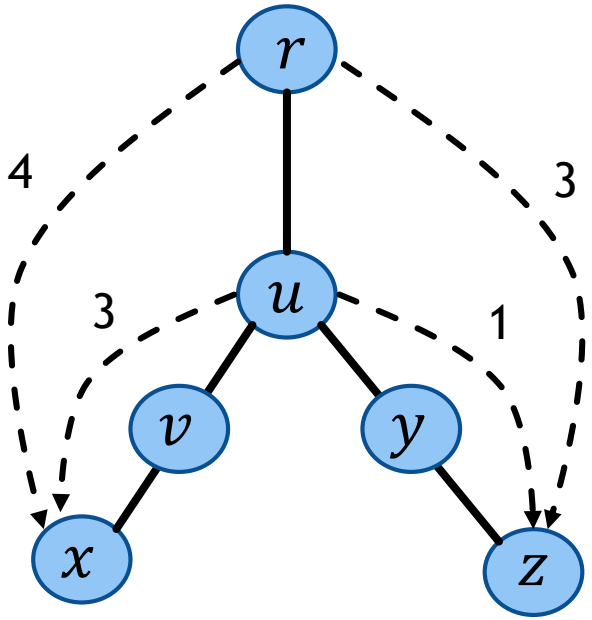
The Algorithm

A vertex v such that $\{v, p(v)\}$ is still not covered, cover it by the edge having weight min_v.



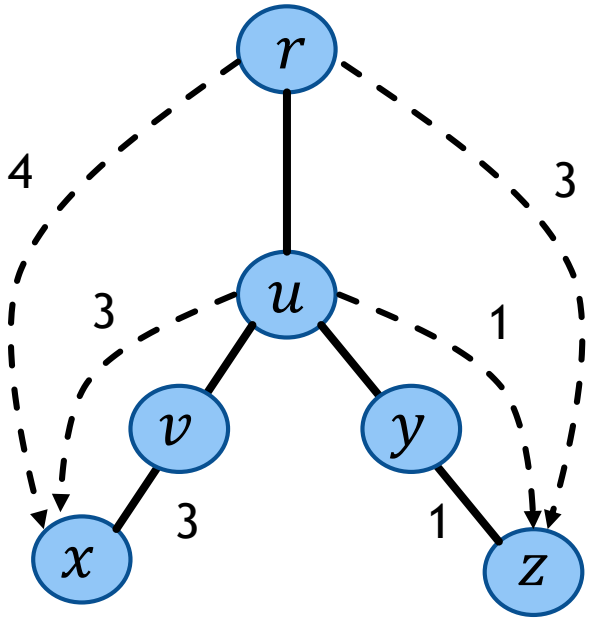
Correctness Proof

We give to a tree edge $\{v, p(v)\}$ the cost min_v.



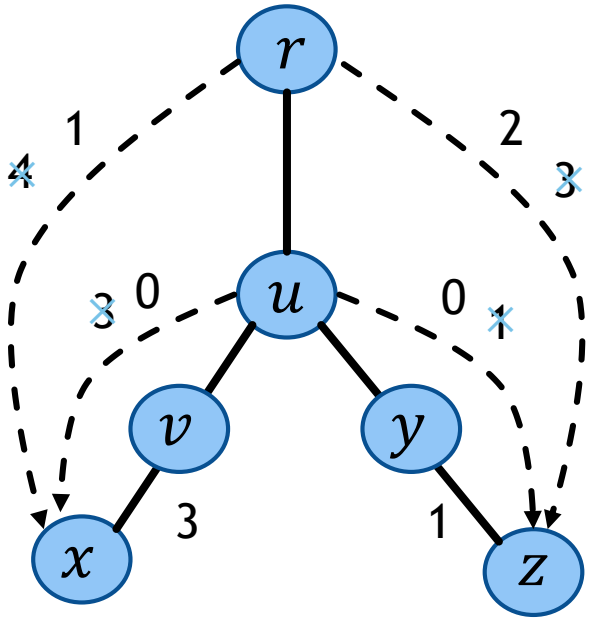
Correctness Proof

We give to a tree edge $\{v, p(v)\}$ the cost min_v.



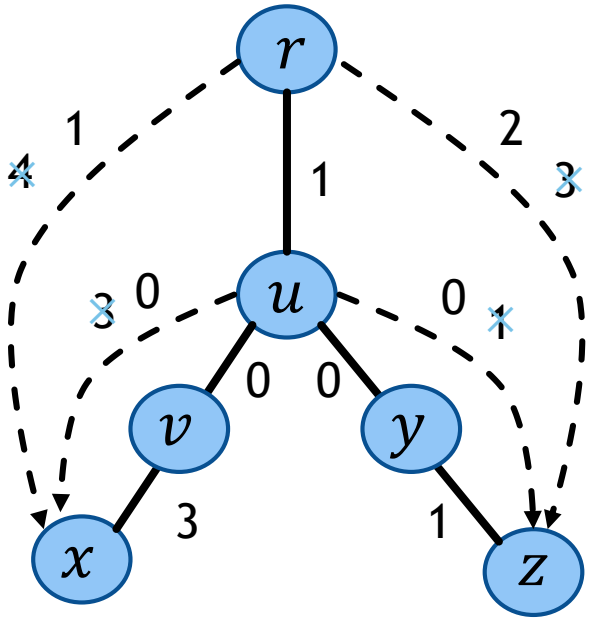
Correctness Proof

We give to a tree edge $\{v, p(v)\}$ the cost min_v.



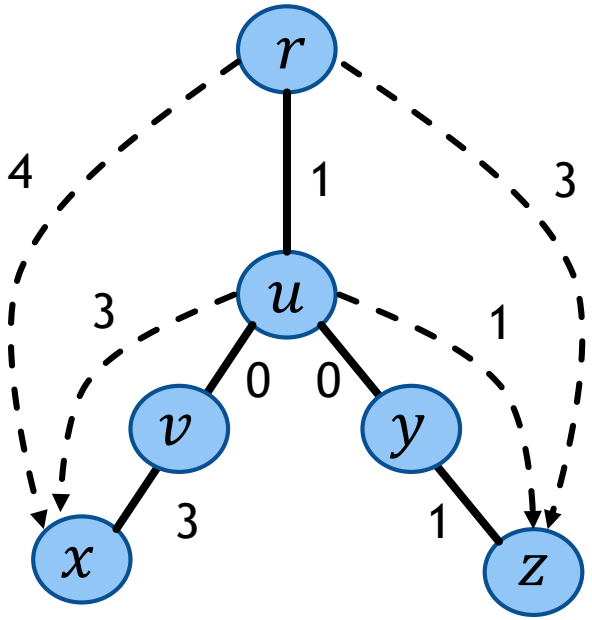
Correctness Proof

We give to a tree edge $\{v, p(v)\}$ the cost min_v.



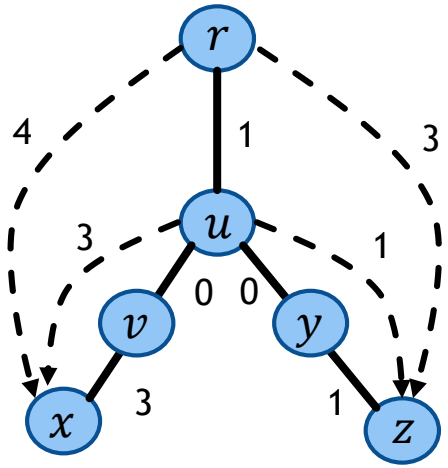
Correctness Proof

We give to a tree edge $\{v, p(v)\}$ the cost min_v.



Correctness Proof

The sum of the costs is equal to the cost of the solution obtained in the algorithm and to the cost of an optimal solution.

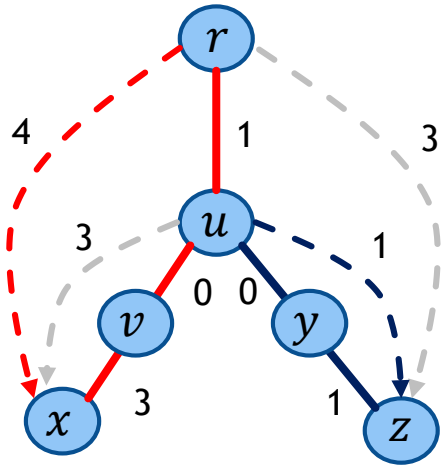


Correctness Proof

- For each edge $e \in A$ we assign a path P_e such that

  $w(e) = \sum_{t \in P_e} c(t)$.

- The paths P_e are disjoint and include all tree edges.



Conclusion

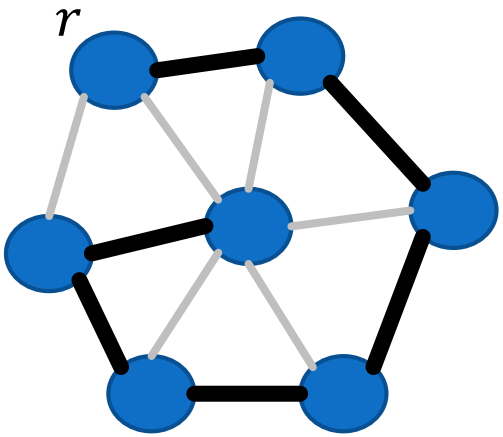
- There is a **2-approximation** for *weighted* TAP in $\boldsymbol{O(h)}$ rounds, where h = height of T.

Is This Optimal?

- TAP is a global problem which requires $\boldsymbol{\Omega(D)}$ rounds, where D = diameter of G
- If $h = O(D)$ our algorithms are optimal up to a constant factor

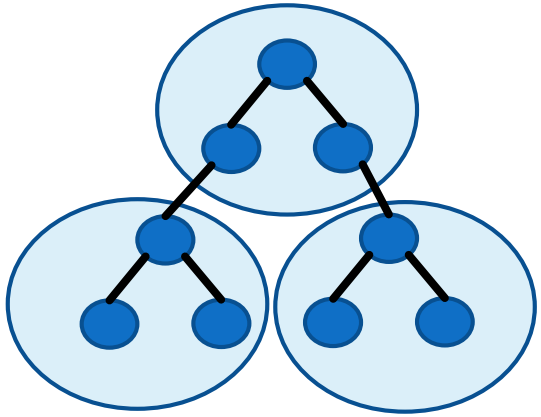
What about the case $h = \omega(D)$?

- We show an $\boldsymbol{\Omega(h)}$ lower bound for *weighted* TAP when $D \approx \log n$, $h \approx \sqrt{n}$



What about the case $h = \omega(D)$?

- We show a **4-approximation** for unweighted TAP in $\boldsymbol{\tilde{O}(\sqrt{n} + D)}$ rounds

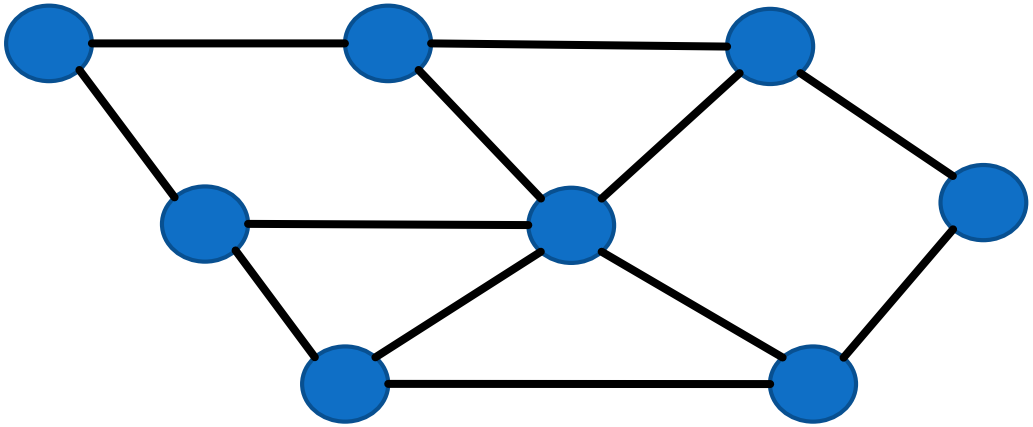


Main Results

- **2-approximation** for unweighted or weighted TAP in $\boldsymbol{O(h)}$ rounds, where h = height of T
- **4-approximation** for unweighted TAP in $\boldsymbol{\tilde{O}(\sqrt{n} + D)}$ rounds, where D = diameter of G
- $\boldsymbol{\Omega(D)}$ lower bound
- $\boldsymbol{\Omega(h)}$ lower bound for weighted TAP when $D \approx \log n$, $h \approx \sqrt{n}$

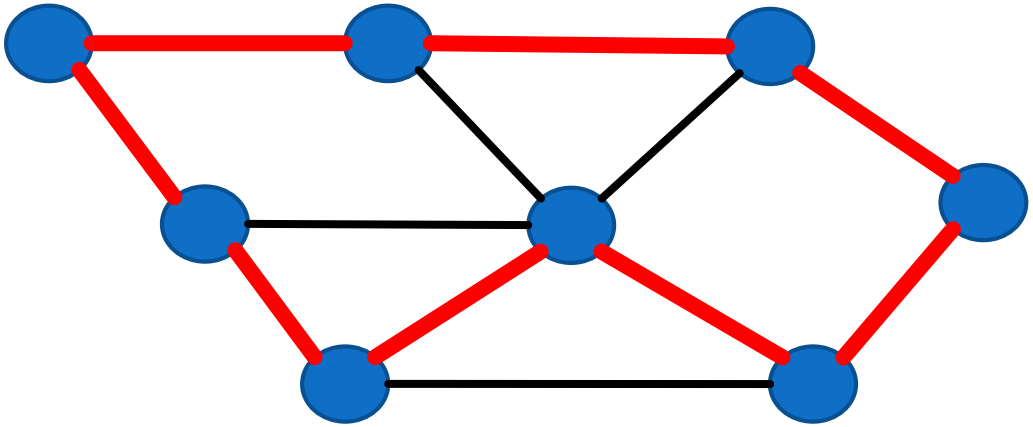
Application: minimum size 2-Edge-Connected Subgraph (2-ECSS)

- Goal: find the minimum size 2-ECSS
- No spanning tree is given



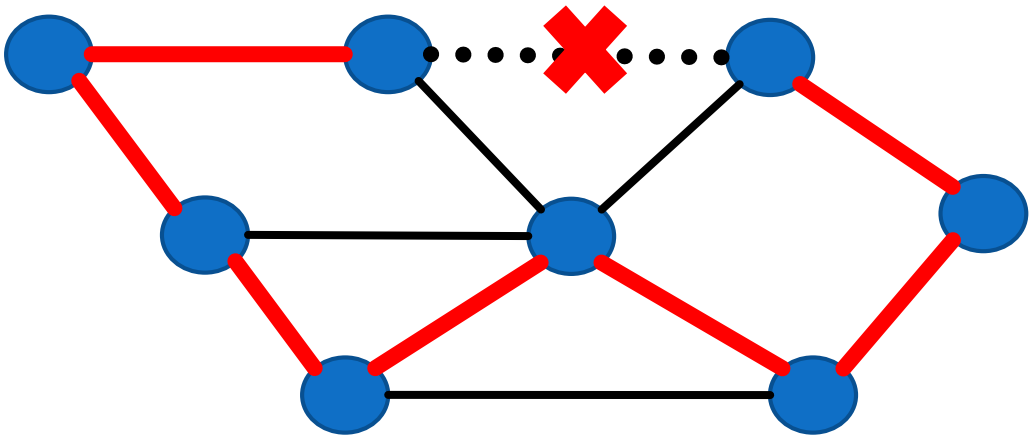
Application: minimum size 2-Edge-Connected Subgraph (2-ECSS)

- Goal: find the minimum size 2-ECSS
- No spanning tree is given



Application: minimum size 2-Edge-Connected Subgraph (2-ECSS)

- Goal: find the minimum size 2-ECSS
- No spanning tree is given



Application: minimum size 2-Edge-Connected Subgraph (2-ECSS)

Previous algorithms:

- $\frac{3}{2}$-approximation in $O(n)$ rounds [Krumke et al. 07]
- 2-approximation in $\tilde{O}(D+\sqrt{n})$ rounds [Thurimella 95]

Application: minimum size 2-Edge-Connected Subgraph (2-ECSS)

Previous algorithms:

- $\frac{3}{2}$-approximation in $O(n)$ rounds [Krumke et al. 07]
- 2-approximation in $\tilde{O}(D+\sqrt{n})$ rounds [Thurimella 95]

Using our TAP algorithm:

- 2-approximation in $O(D)$ rounds

Application: minimum ***weight*** 2-Edge-Connected Subgraph (2-ECSS)

Previous algorithms:

- 3-approximation in $O(n \log n)$ rounds

  [Krumke et al. 07]

Using our TAP algorithm:

- 3-approximation in $\tilde{O}(h_{MST} + \sqrt{n})$ rounds
- Lower bound of $\tilde{\Omega}(D + \sqrt{n})$ rounds for any polynomial approximation

Future Work

- Design efficient algorithms for *weighted* TAP and *weighted* 2-ECSS.
- Design **distributed** algorithms for additional **connectivity** problems:
  - Higher connectivity
  - Vertex connectivity

Can we improve the round complexity?

- We show an $\tilde{O}(D + \sqrt{n})$-round $O(\log n)$-approximation for weighted TAP and weighted 2-ECSS.

The Algorithm

- TAP is a set cover problem where the goal is to cover all tree edges by non-tree edges.
- The cost-effectiveness of a non-tree edge e is $\frac{C_e}{w(e)}$ where C_e is the number of uncovered tree edges covered by e.

Sequential Greedy Algorithm

- At each step, add to the augmentation the edge with maximal cost-effectiveness.
- Continue until all the tree edges are covered.

Distributed Algorithm

- We need a mechanism for symmetry breaking.
- We need to do many global computations in parallel.

The Algorithm

- At each step, find all the edges with maximal cost-effectiveness, they are the *candidates*.
- Each candidate edge chooses a random number $r \in [0,1]$.
- Each uncovered tree edge votes to the first candidate that covers it.
- An edge is added to the augmentation if it gets at least $\frac{1}{8}$ of the votes of the tree edges it covers.
- We continue until all tree edges are covered.

The Algorithm

- We need to do many global computations in parallel: computing cost-effectiveness, computing the number of votes…
- To achieve this, we decompose the tree into fragments, following a decomposition presented for solving the fault-tolerant MST problem [Ghaffari and Parter, 2016].

Higher Connectivity

- What about minimum k-ECSS?
- The same basic approach may be useful also for k-ECSS.
- However, now the cost-effectiveness of an edge depends on the number of cuts it covers.