Better-Than-2 Approximations for Weighted Tree Augmentation

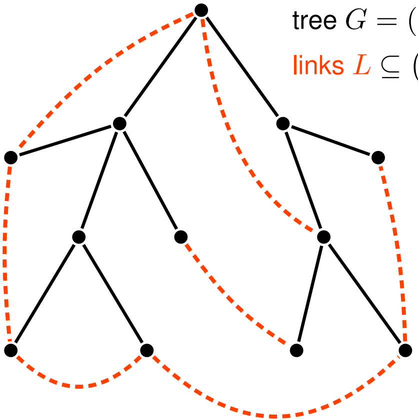
Vera Traub

ETH Zürich

Rico Zenklusen

ETH Zürich

Weighted Tree Augmentation (WTAP)



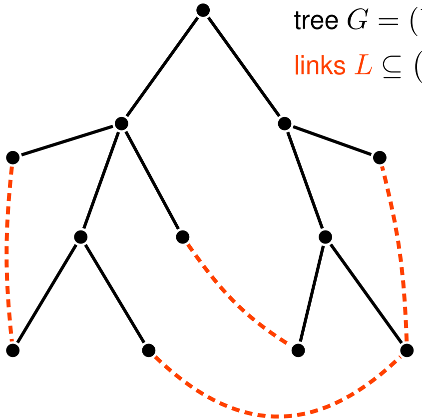
tree $G = (V, E)$

links $L \subseteq \binom{V}{2}$ with weights $w : L \to \mathbb{R}_{>0}$

WTAP

Find a min weight set $F \subseteq L$ of links s.t.
G becomes 2-edge-connected when adding F.

Weighted Tree Augmentation (WTAP)



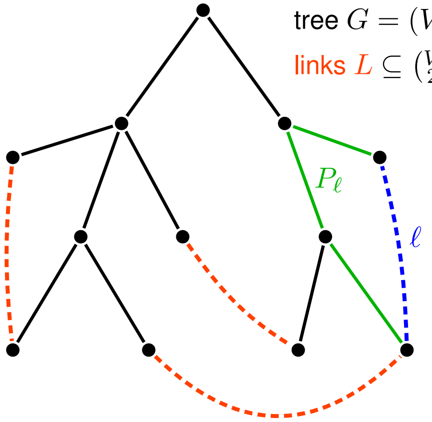
tree $G = (V, E)$

links $L \subseteq \binom{V}{2}$ with weights $w : L \to \mathbb{R}_{>0}$

WTAP

Find a min weight set $F \subseteq L$ of links s.t.
G becomes 2-edge-connected when adding F.

Weighted Tree Augmentation (WTAP)



tree $G = (V, E)$

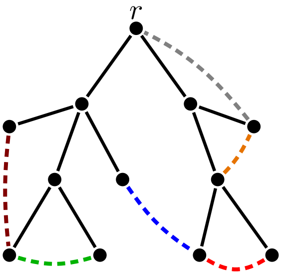
links $L \subseteq \binom{V}{2}$ with weights $w : L \to \mathbb{R}_{>0}$

WTAP

Find a min weight set $F \subseteq L$ of links s.t.
G becomes 2-edge-connected when adding F.

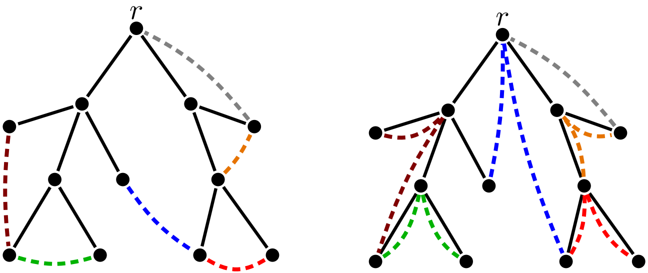
Equivalent:
Every edge $e \in E$ must be covered by a link $\ell \in F$,
i.e., $e \in P_\ell$ for some $\ell \in F$.

Warm-up: a simple 2-approximation



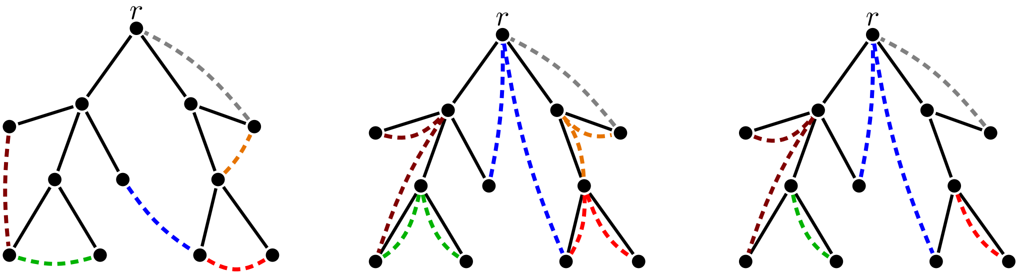
① Pick an arbitrary root $r \in V$.

Warm-up: a simple 2-approximation



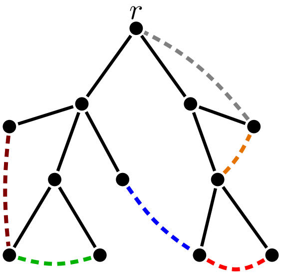
1. Pick an arbitrary root $r \in V$.
2. "Split" every link ℓ into two up-links, each with weight $w(\ell)$.

Warm-up: a simple 2-approximation



1. Pick an arbitrary root $r \in V$.
2. "Split" every link ℓ into two up-links, each with weight $w(\ell)$.
3. Compute an optimal up-link solution.

Warm-up: a simple 2-approximation



1. Pick an arbitrary root $r \in V$.
2. "Split" every link ℓ into two up-links, each with weight $w(\ell)$.
3. Compute an optimal up-link solution.

solve natural LP (integral), or use dynamic programming

Better-than-2 approximations for special cases

- **unweighted tree augmentation (TAP):** 1.393-approximation [Cecchetto, T., Zenklusen, 2021]

  (improving on [Nagamochi, 2003], [Even, Feldmann, Kortsarz, Nutov, 2009], [Cheriyan, Gao, 2018], [Kortaz, Nutov, 2016], [Kortaz, Nutov, 2018], [Adjashvilli, 2018], [Nutov, 2017], [Fiorini, Groß, Könemann, Sanità, 2018], [Grandoni, Kalaitzis, Zenklusen, 2018])

- **bounded-diameter trees:** $(1 + \ln 2)$-approximation [Cohen, Nutov, 2013]

- better-than-2 approximation if an opt. solution to natural LP has no small fractional values [Iglesias, Ravi, 2018]

Our result

Theorem

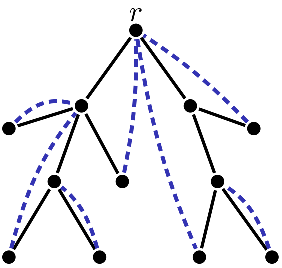
There is a $(1.5+\varepsilon)$-approximation algorithm for Weighted Tree Augmentation (WTAP) for any fixed $\varepsilon > 0$.

Outline of this talk:

1. relative greedy algorithm: $(1+\ln 2+\varepsilon)$-approximation
2. local search algorithm: $(1.5+\varepsilon)$-approximation
3. main technical ingredient: decomposition theorem

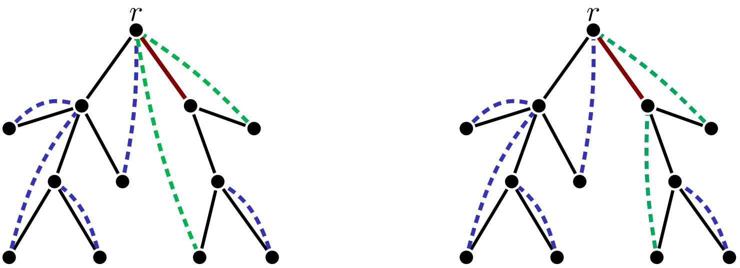
The Relative Greedy Algorithm

The starting solution for relative greedy



① Compute optimal up-link solution U (2-approximation).

The starting solution for relative greedy

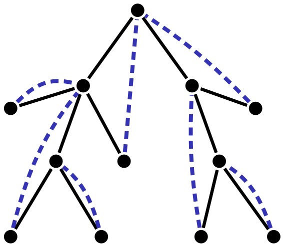


1. Compute optimal up-link solution U (2-approximation).
2. "Shorten" up-links s.t. P_u with $u \in U$ are disjoint, i.e., every edge is covered by exactly one link.

Relative greedy

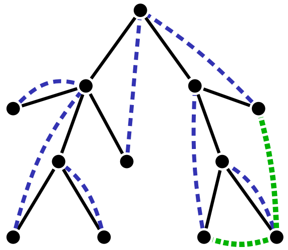
Invariant: $U \cup F$ is a WTAP solution

(1) $U :=$ 2-approximate up-link solution s.t. the paths P_u with $u \in U$ are disjoint.
$F := \emptyset$



Relative greedy

Invariant: $U \cup F$ is a WTAP solution



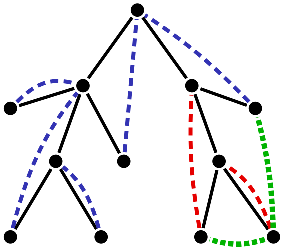
① $U :=$ 2-approximate up-link solution s.t. the paths P_u with $u \in U$ are disjoint.
$F := \emptyset$

② As long as $w(U \cup F)$ decreases:

- Select a component $C \subseteq L$.
- Add C to F.

Relative greedy

Invariant: $U \cup F$ is a WTAP solution



① U := 2-approximate up-link solution s.t. the paths P_u with $u \in U$ are disjoint.
$F := \emptyset$

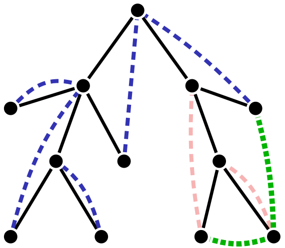
② As long as $w(U \cup F)$ decreases:

- Select a component $C \subseteq L$.
- Add C to F.
- Remove the following from U:

$$\mathrm{Drop}_U(C) := \left\{ u \in U : P_u \subseteq \bigcup_{\ell \in C} P_\ell \right\}$$

Relative greedy

Invariant: $U \cup F$ is a WTAP solution



1. U := 2-approximate up-link solution s.t. the paths P_u with $u \in U$ are disjoint.
   $F := \emptyset$

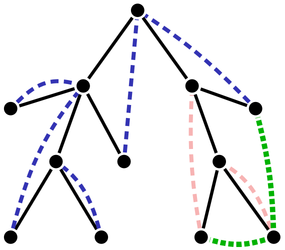
2. As long as $w(U \cup F)$ decreases:
   - Select a component $C \subseteq L$.
   - Add C to F.
   - Remove the following from U:

$$\text{Drop}_U(C) := \left\{ u \in U : P_u \subseteq \bigcup_{\ell \in C} P_\ell \right\}$$

3. Return $U \cup F$.

Relative greedy

Invariant: $U \cup F$ is a WTAP solution



1. U := 2-approximate up-link solution s.t. the paths P_u with $u \in U$ are disjoint.
   $F := \emptyset$

2. As long as $w(U \cup F)$ decreases:
   - Select a component $C \subseteq L$.
   - Add C to F.
   - Remove the following from U:

$$\mathrm{Drop}_U(C) := \left\{ u \in U : P_u \subseteq \bigcup_{\ell \in C} P_\ell \right\}$$

3. Return $U \cup F$.

Choose C s.t. it minimizes

$$\frac{w(C)}{w(\mathrm{Drop}_U(C))}$$

among a restricted class of components.

How should we define components?

We need:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\mathrm{Drop}_U(C))}$.

How should we define components?

We need:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\mathrm{Drop}_U(C))}$.

(b) If $w(U) \gg w(\mathrm{OPT})$, there exist a component C with $\frac{w(C)}{w(\mathrm{Drop}_U(C))} \ll 1$.

How should we define components?

We need:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\text{Drop}_U(C))}$.

(b) If $w(U) \gg w(\text{OPT})$, there exist a component C with $\frac{w(C)}{w(\text{Drop}_U(C))} \ll 1$.

constant size link sets

(a) ✓ (enumerate)

(b) ✗

How should we define components?

We need:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\mathrm{Drop}_U(C))}$.

(b) If $w(U) \gg w(\mathrm{OPT})$, there exist a component C with $\frac{w(C)}{w(\mathrm{Drop}_U(C))} \ll 1$.

constant size link sets

(a) ✓ (enumerate)

(b) ✗

arbitrary link sets

(a) ✗

(b) ✓ (for $C = \mathrm{OPT}$)

How should we define components?

We need:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\mathrm{Drop}_U(C))}$.

(b) If $w(U) \gg w(\mathrm{OPT})$, there exist a component C with $\frac{w(C)}{w(\mathrm{Drop}_U(C))} \ll 1$.

constant size link sets

(a) ✓ (enumerate)

(b) ✗

???

(a) ✓

(b) ✓

arbitrary link sets

(a) ✗

(b) ✓ (for $C = \mathrm{OPT}$)

How should we define components?

We need:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\mathrm{Drop}_U(C))}$.

(b) If $w(U) \gg w(\mathrm{OPT})$, there exist a component C with $\frac{w(C)}{w(\mathrm{Drop}_U(C))} \ll 1$.

constant size link sets

(a) ✓ (enumerate)

(b) ✗

k-thin link sets

(a) ✓

(b) ✓

arbitrary link sets

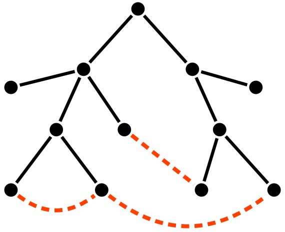
(a) ✗

(b) ✓ (for $C = \mathrm{OPT}$)

k-thin components

Definition

$C \subseteq L$ is <u>k-thin</u> if for every $v \in V$, there are at most k links $\ell \in C$ for which v lies on P_ℓ.

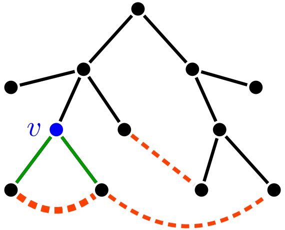


2-thin component

k-thin components

Definition

$C \subseteq L$ is k-thin if for every $v \in V$,
there are at most k links $\ell \in C$
for which v lies on P_ℓ.

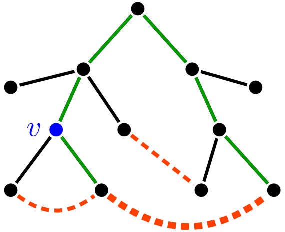


2-thin component

k-thin components

Definition

$C \subseteq L$ is <u>k-thin</u> if for every $v \in V$, there are at most k links $\ell \in C$ for which v lies on P_ℓ.

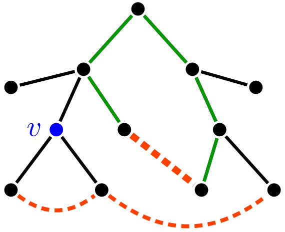


2-thin component

k-thin components

Definition

$C \subseteq L$ is $\underline{k\text{-thin}}$ if for every $v \in V$, there are at most k links $\ell \in C$ for which v lies on P_ℓ.

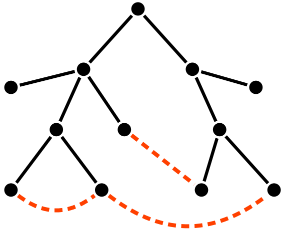


2-thin component

k-thin components

Definition

$C \subseteq L$ is k-thin if for every $v \in V$,
there are at most k links $\ell \in C$
for which v lies on P_ℓ.



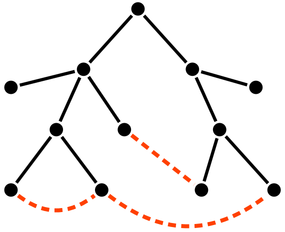
Then:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\mathrm{Drop}_U(C))}$. ✓ (dynamic program)

k-thin components

Definition

$C \subseteq L$ is k-thin if for every $v \in V$,
there are at most k links $\ell \in C$
for which v lies on P_ℓ.



2-thin component

Then:

(a) We can efficiently find a component C minimizing $\frac{w(C)}{w(\text{Drop}_U(C))}$. ✓ (dynamic program)

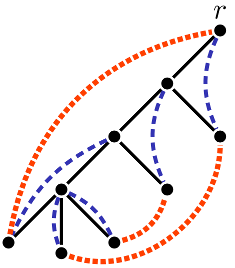
(b) If $w(U) \gg w(\text{OPT})$, there exist a component C with $\frac{w(C)}{w(\text{Drop}_U(C))} \ll 1$. ✓

(decomposition theorem)

The decomposition theorem

Fix $\varepsilon > 0$.

U := set of up-links s.t. the paths P_u with $u \in U$ are disjoint.



Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil 1/\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \geq (1-\varepsilon) \cdot w(U).$$

The approximation ratio of relative greedy

Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil 1/\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \;\geq\; (1-\varepsilon) \cdot w(U).$$

Proving (b):

The approximation ratio of relative greedy

Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil 1/\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \;\geq\; (1-\varepsilon) \cdot w(U).$$

Proving (b): If $w(U) \gg w(\mathrm{OPT})$,

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \;\gg\; w(\mathrm{OPT}) \;=\; \sum_{C \in \mathcal{C}} w(C).$$

$\Longrightarrow$ There exist a component C with $\frac{w(C)}{w(\mathrm{Drop}_U(C))} \ll 1$.

The approximation ratio of relative greedy

Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil 1/\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \;\geq\; (1-\varepsilon) \cdot w(U).$$

Proving (b): If $w(U) \gg w(\mathrm{OPT})$,

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \;\gg\; w(\mathrm{OPT}) \;=\; \sum_{C \in \mathcal{C}} w(C).$$

$\Longrightarrow$ There exist a component C with $\frac{w(C)}{w(\mathrm{Drop}_U(C))} \ll 1$.

Theorem

The relative greedy algorithm for WTAP has approximation ratio $1 + \ln 2 + \varepsilon < 1.7$.

Local Search

Improving on the Relative Greedy Algorithm

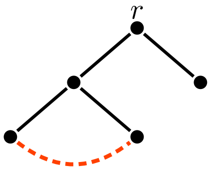
Relative greedy: Replace only up-links from the starting solution.

Now: We want to gain also on links added in previous iterations.

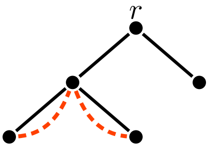
Improving on the Relative Greedy Algorithm

Relative greedy: Replace only up-links from the starting solution.

Now: We want to gain also on links added in previous iterations.



link ℓ
(added in earlier iteration)



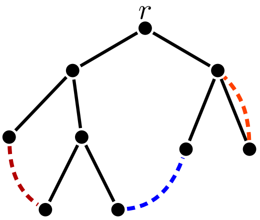
witness set W_ℓ

Key idea

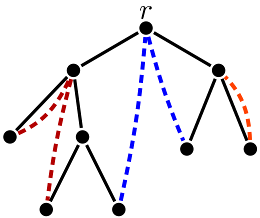
Reward partial progress, i.e., covering one of the up-links in W_ℓ.

Rewarding partial progress

WTAP solution F



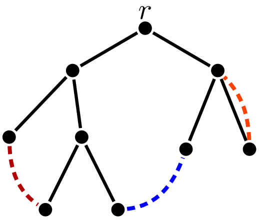
up-link solution $U = \dot{\bigcup}_{\ell \in F} W_\ell$



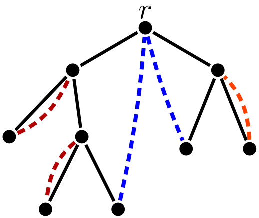
- If an up-link in $W_\ell \subseteq U$ is covered by a new component C, remove it.
- If W_ℓ is empty, remove ℓ from F.

Rewarding partial progress

WTAP solution F



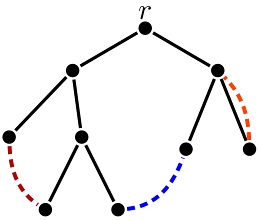
up-link solution $U = \dot{\bigcup}_{\ell \in F} W_\ell$



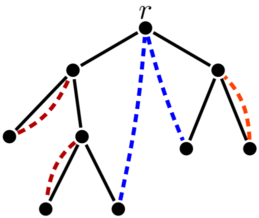
- If an up-link in $W_\ell \subseteq U$ is covered by a new component C, remove it.
- If W_ℓ is empty, remove ℓ from F.

Rewarding partial progress

WTAP solution F



up-link solution $U = \dot\bigcup_{\ell \in F} W_\ell$

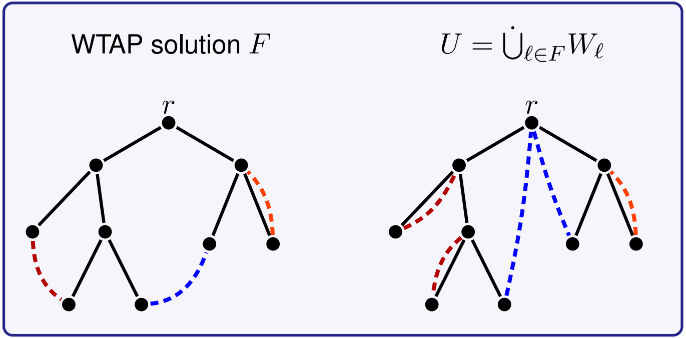


- If an up-link in $W_\ell \subseteq U$ is covered by a new component C, remove it.
- If W_ℓ is empty, remove ℓ from F.
- Minimize the potential

$$\Phi(F) := \sum_{\ell \in F : |W_\ell| = 1} w(\ell) + \sum_{\ell \in F : |W_\ell| = 2} \frac{3}{2} \cdot w(\ell)$$

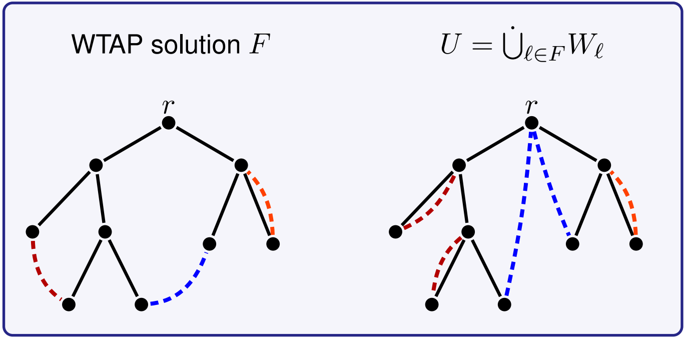
The potential function Φ

$$\Phi(F) := \sum_{\substack{\ell \in F: \\ |W_\ell|=1}} w(\ell) + \sum_{\substack{\ell \in F: \\ |W_\ell|=2}} \tfrac{3}{2} \cdot w(\ell)$$



The potential function Φ

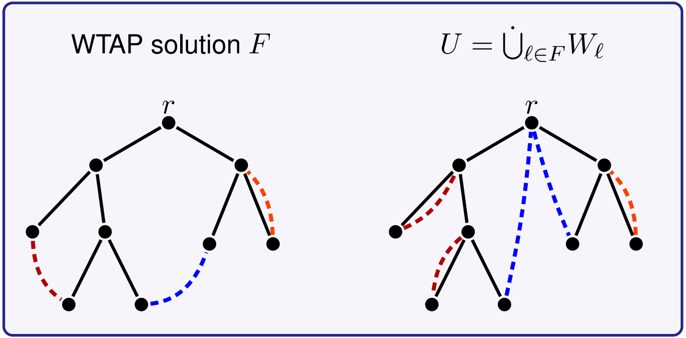
$$\Phi(F) := \sum_{\substack{\ell \in F: \\ |W_\ell|=1}} w(\ell) + \sum_{\substack{\ell \in F: \\ |W_\ell|=2}} \frac{3}{2} \cdot w(\ell)$$



- When adding ℓ to F (and W_ℓ to U), the potential increases by at most $\frac{3}{2} \cdot w(\ell)$.

The potential function Φ

$$\Phi(F) := \sum_{\substack{\ell \in F: \\ |W_\ell|=1}} w(\ell) + \sum_{\substack{\ell \in F: \\ |W_\ell|=2}} \frac{3}{2} \cdot w(\ell)$$

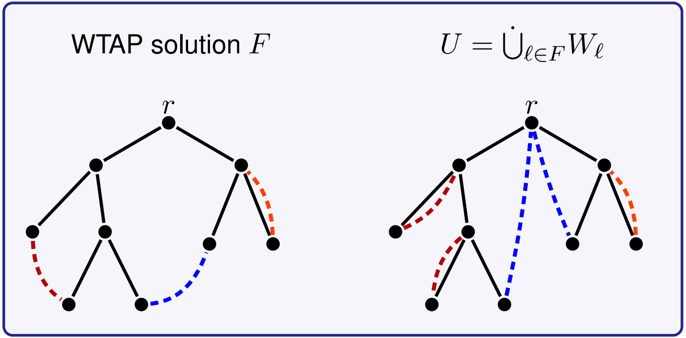


- When adding ℓ to F (and W_ℓ to U), the potential increases by at most $\frac{3}{2} \cdot w(\ell)$.
- When removing $u \in W_\ell$, the potential decreases by

$$\overline{w}(u) := \begin{cases} w(\ell) & \text{if } |W_\ell| = 1 \\ \frac{1}{2} \cdot w(\ell) & \text{if } |W_\ell| = 2 \end{cases}$$

The potential function Φ

$$\Phi(F) := \sum_{\substack{\ell \in F: \\ |W_\ell|=1}} w(\ell) + \sum_{\substack{\ell \in F: \\ |W_\ell|=2}} \frac{3}{2} \cdot w(\ell)$$



- When adding ℓ to F (and W_ℓ to U), the potential increases by at most $\frac{3}{2} \cdot w(\ell)$.
- When removing $u \in W_\ell$, the potential decreases by

$$\overline{w}(u) := \begin{cases} w(\ell) & \text{if } |W_\ell| = 1 \\ \frac{1}{2} \cdot w(\ell) & \text{if } |W_\ell| = 2 \end{cases}$$

Observation

$$\overline{w}(U) = w(F)$$

Local minima are good approximations

A Local Search Step with component C

- When adding C to F
  (and the corresponding witness sets to U),

  $\Phi(F)$ increases by at most $\frac{3}{2} \cdot w(C)$.

- When removing $\mathrm{Drop}_U(C)$ from U,

  $\Phi(F)$ decreases by at least $\overline{w}(\mathrm{Drop}_U(C))$.

Local minima are good approximations

A Local Search Step with component C

- When adding C to F
  (and the corresponding witness sets to U),

  $\Phi(F)$ increases by at most $\frac{3}{2} \cdot w(C)$.

- When removing $\mathrm{Drop}_U(C)$ from U,

  $\Phi(F)$ decreases by at least $\overline{w}(\mathrm{Drop}_U(C))$.

Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil {}^1\!/\!_\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} \overline{w}(\mathrm{Drop}_U(C)) \geq (1-\varepsilon) \cdot \overline{w}(U)$$

$$= (1-\varepsilon) \cdot w(F).$$

Local minima are good approximations

A Local Search Step with component C

- When adding C to F (and the corresponding witness sets to U),

  $\Phi(F)$ increases by at most $\frac{3}{2} \cdot w(C)$.

- When removing $\mathrm{Drop}_U(C)$ from U,

  $\Phi(F)$ decreases by at least $\overline{w}(\mathrm{Drop}_U(C))$.

Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil 1/\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} \overline{w}(\mathrm{Drop}_U(C)) \geq (1-\varepsilon) \cdot \overline{w}(U) = (1-\varepsilon) \cdot w(F).$$

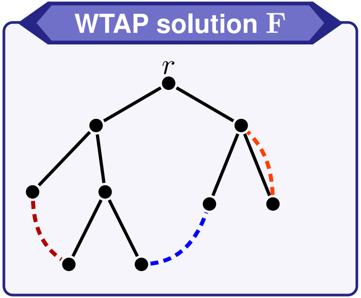
If $w(F) \gg \frac{3}{2} \cdot w(\mathrm{OPT})$,

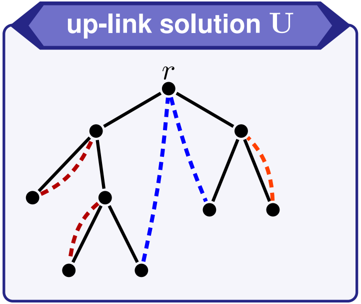
$$\sum_{C \in \mathcal{C}} \overline{w}(\mathrm{Drop}_U(C)) \gg \frac{3}{2} \cdot w(\mathrm{OPT}) = \sum_{C \in \mathcal{C}} \frac{3}{2} \cdot w(C).$$

$\Longrightarrow$ There exists an improving component!

Local search algorithm

1. F := arbitrary WTAP solution
   U := corresponding up-link solution

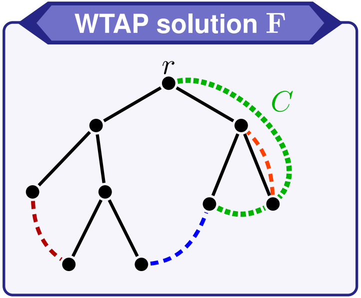


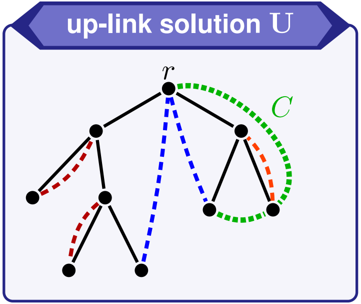


Local search algorithm

1. F := arbitrary WTAP solution
   U := corresponding up-link solution

2. As long as $\Phi(F)$ improves significantly:
   - $C := k$-thin component maximizing $\overline{w}(\mathrm{Drop}_U(C)) - 1.5 \cdot w(C)$.



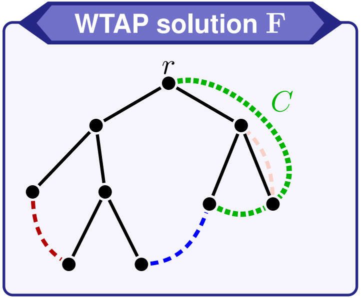


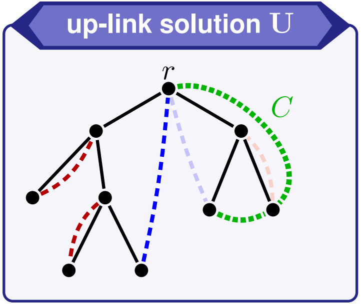
Local search algorithm

① F := arbitrary WTAP solution
U := corresponding up-link solution

② As long as $\Phi(F)$ improves significantly:

- C := k-thin component maximizing $\overline{w}(\mathrm{Drop}_U(C)) - 1.5 \cdot w(C)$.
- Remove $\mathrm{Drop}_U(C)$ from U.
  If a witness set W_ℓ became empty, remove ℓ from F.

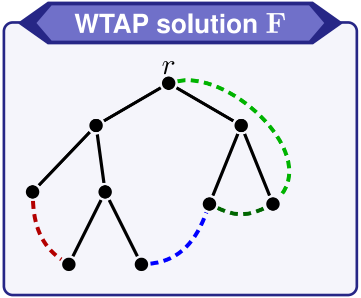


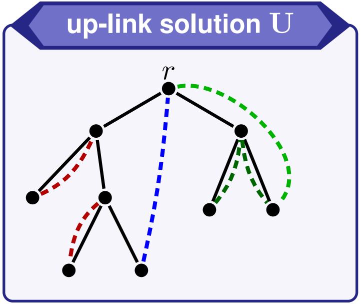


Local search algorithm

1. F := arbitrary WTAP solution
   U := corresponding up-link solution

2. As long as $\Phi(F)$ improves significantly:
   - C := k-thin component maximizing $\overline{w}(\mathrm{Drop}_U(C)) - 1.5 \cdot w(C)$.
   - Remove $\mathrm{Drop}_U(C)$ from U.
     If a witness set W_ℓ became empty, remove ℓ from F.
   - Add C to F and the corresponding witness sets to U.



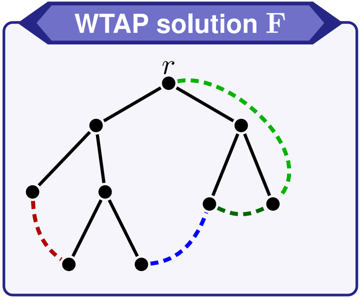


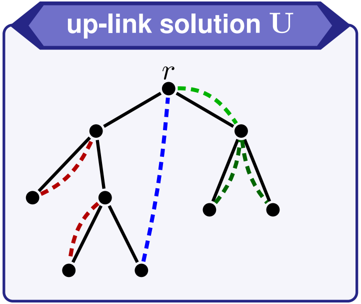
Local search algorithm

1. $F :=$ arbitrary WTAP solution
   $U :=$ corresponding up-link solution

2. As long as $\Phi(F)$ improves significantly:
   - $C := k$-thin component maximizing $\overline{w}(\mathrm{Drop}_U(C)) - 1.5 \cdot w(C)$.
   - Remove $\mathrm{Drop}_U(C)$ from U.
     If a witness set W_ℓ became empty, remove ℓ from F.
   - Add C to F and the corresponding witness sets to U.
   - ”Shorten” up-links.

3. Return F.





Local search algorithm

1. $F :=$ arbitrary WTAP solution
   $U :=$ corresponding up-link solution

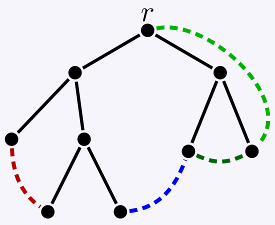
2. As long as $\Phi(F)$ improves significantly:
   - $C := k$-thin component maximizing $\overline{w}(\mathrm{Drop}_U(C)) - 1.5 \cdot w(C)$.
   - Remove $\mathrm{Drop}_U(C)$ from U.
     If a witness set W_ℓ became empty, remove ℓ from F.
   - Add C to F and the corresponding witness sets to U.
   - "Shorten" up-links.

3. Return F.

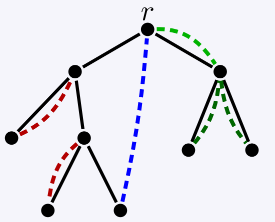
Theorem

The above algorithm is a $(1.5+\varepsilon)$-approximation algorithm for Weighted Tree Augmentation.

WTAP solution F

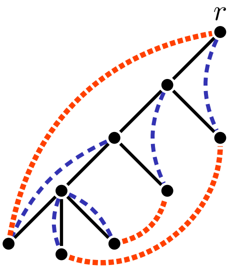


up-link solution U



Proving the Decomposition Theorem

Proving the decomposition theorem



We have

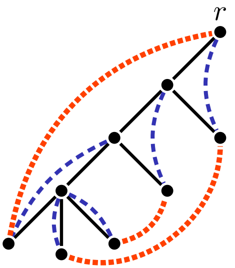
- a set U of up-links s.t. the paths P_u with $u \in U$ are disjoint,
- WTAP solution OPT, and
- constants $\varepsilon > 0$ and $k := \lceil \frac{1}{\varepsilon} \rceil$.

Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into k-thin components s.t.:

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \;\geq\; (1 - \varepsilon) \cdot w(U).$$

Proving the decomposition theorem



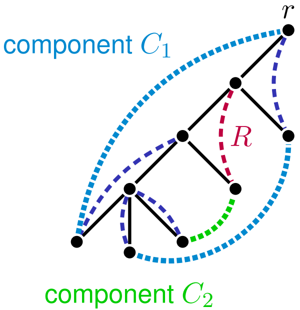
We have

- a set U of up-links s.t. the paths P_u with $u \in U$ are disjoint,
- WTAP solution OPT, and
- constants $\varepsilon > 0$ and $k := \lceil \frac{1}{\varepsilon} \rceil$.

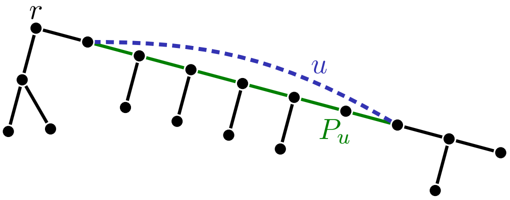
Goal

- Select "uncovered" up-links $R \subseteq U$ with $w(R) \leq \varepsilon \cdot w(U)$.
- Construct partition $\mathcal{C}$ of OPT into k-thin components s.t. all up-links in $U \setminus R$ are covered, i.e.,

$$U \setminus R \subseteq \bigcup_{C \in \mathcal{C}} \mathrm{Drop}_U(C)$$

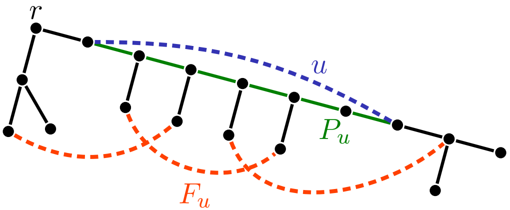


Proof Outline



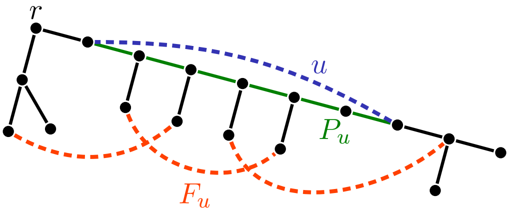
1. For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.

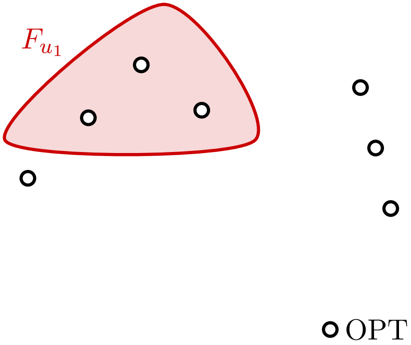
Proof Outline



1. For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.

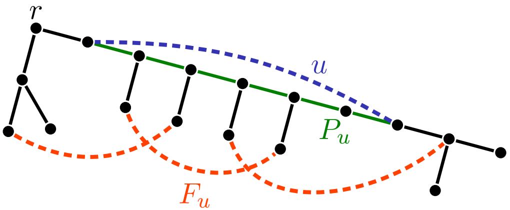
Proof Outline

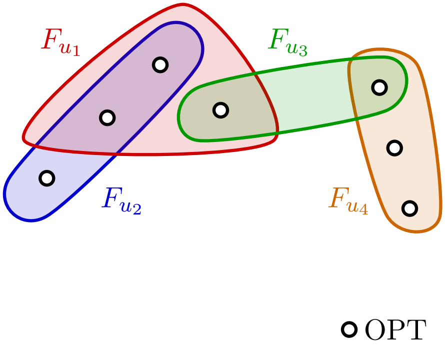




① For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.

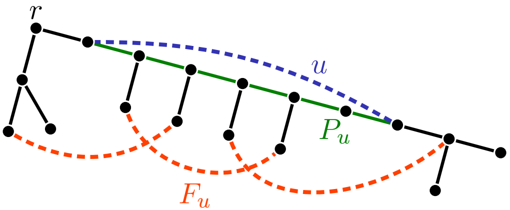
Proof Outline

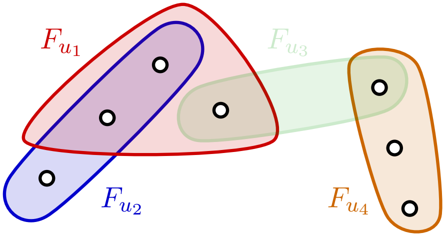




1. For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.

Proof Outline



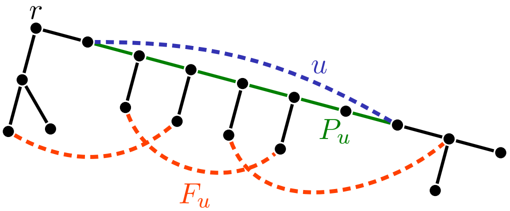


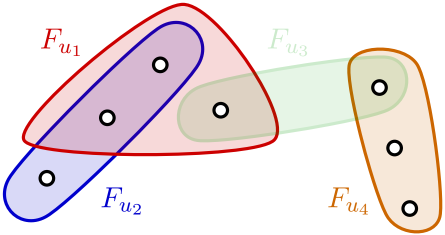
1. For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.
2. Select $R \subseteq U$ with $w(R) \leq \varepsilon \cdot w(U)$.

○ OPT

$R = \{u_3\}$

Proof Outline



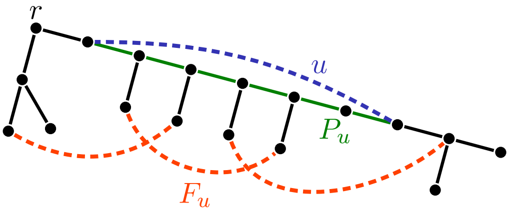


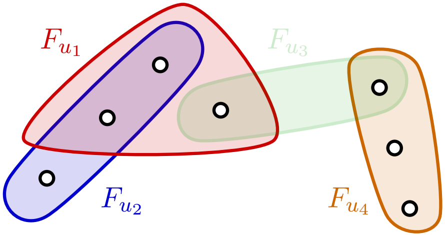
1. For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.
2. Select $R \subseteq U$ with $w(R) \leq \varepsilon \cdot w(U)$.
3. Partition OPT into components s.t.
   for all $u \in U \setminus R$, there is a component C with $F_u \subseteq C$.

○ OPT

$R = \{u_3\}$

Proof Outline





1. For $u \in U$, fix a covering $F_u \subseteq \mathrm{OPT}$ of P_u.
2. Select $R \subseteq U$ with $w(R) \leq \varepsilon \cdot w(U)$.
3. Partition OPT into components s.t.
   for all $u \in U \setminus R$, there is a component C with $F_u \subseteq C$.

○ OPT

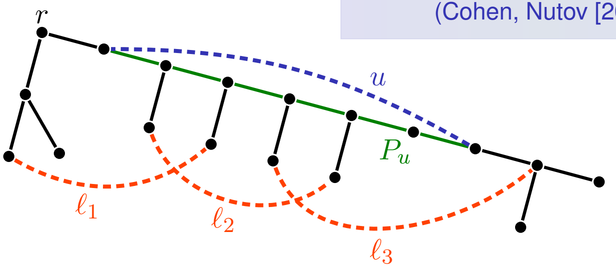
$R = \{u_3\}$

Challenge

Make choices in (1) and (2) s.t. the resulting components are k-thin.

The Dependency Graph

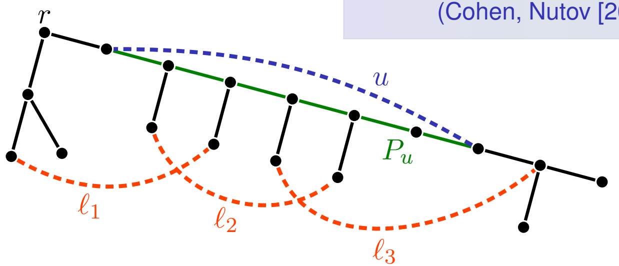
(Cohen, Nutov [2013])



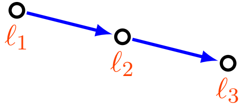
minimal covering F_u of P_u.

The Dependency Graph

(Cohen, Nutov [2013])



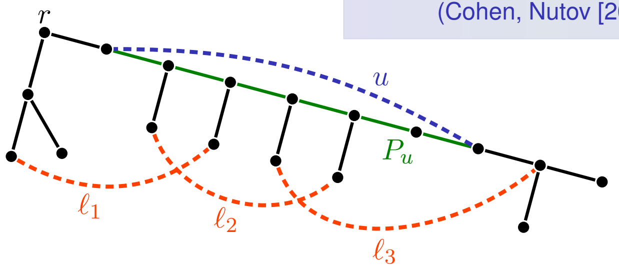
minimal covering F_u of P_u.



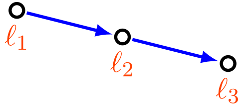
directed path with vertex set F_u and arc set A_u

The Dependency Graph

(Cohen, Nutov [2013])



minimal covering F_u of P_u.

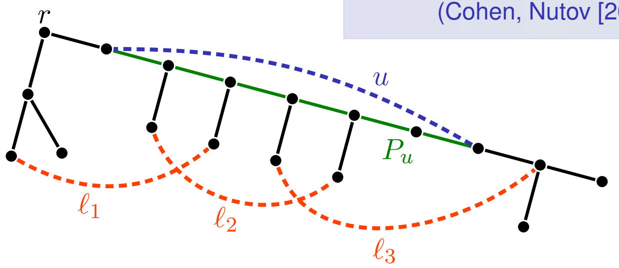


directed path with vertex set F_u and arc set A_u

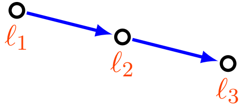
dependency graph for U $:=$ digraph with vertex set OPT and arc set $\dot\bigcup_{u \in U} A_u$.

The Dependency Graph

(Cohen, Nutov [2013])



minimal covering F_u of P_u.



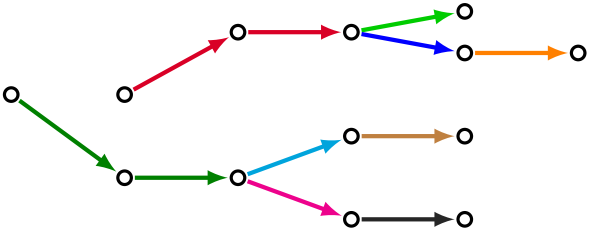
directed path with vertex set F_u and arc set A_u

dependency graph for U $:=$ digraph with vertex set OPT and arc set $\dot{\bigcup}_{u \in U} A_u$.

The dependency graph is a branching.

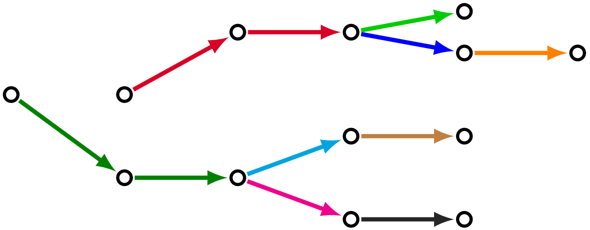
Thinness and the Dependency Graph

different colors = different paths A_u



Thinness and the Dependency Graph

different colors = different paths A_u



Key properties of the dependency graph

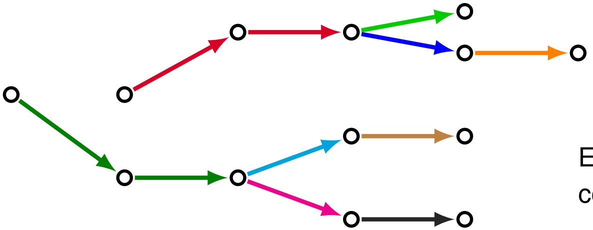
For a careful choice of the coverings F_u:

(i) The dependency graph is a branching.

(ii) If every path in the dependency graph intersects $\leq k-1$ sets A_u, then every component is k-thin.

Thinness and the Dependency Graph

different colors = different paths A_u



Every connected component corresponds to a 4-thin link set.

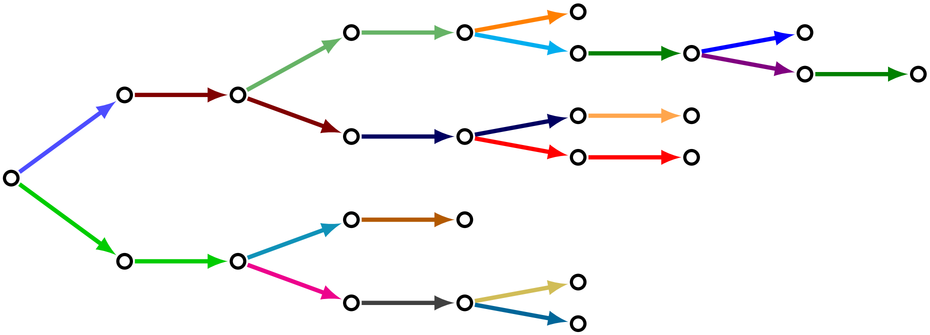
Key properties of the dependency graph

For a careful choice of the coverings F_u:

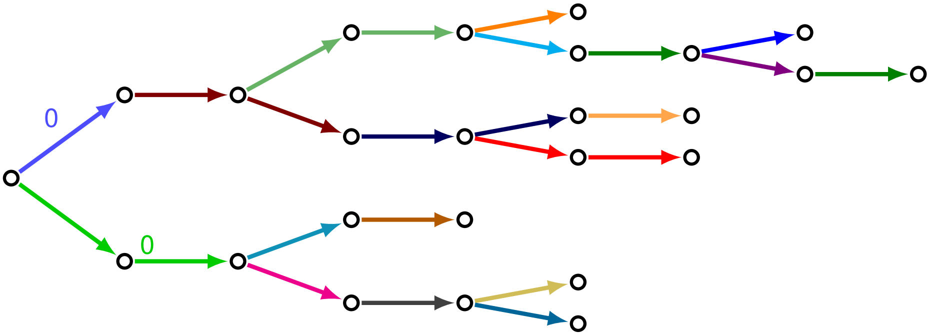
(i) The dependency graph is a branching.

(ii) If every path in the dependency graph intersects $\leq k-1$ sets A_u, then every component is k-thin.

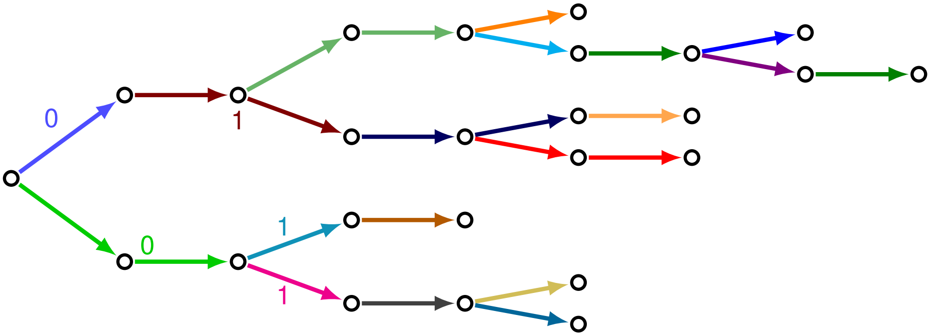
Selecting the uncovered up-links R



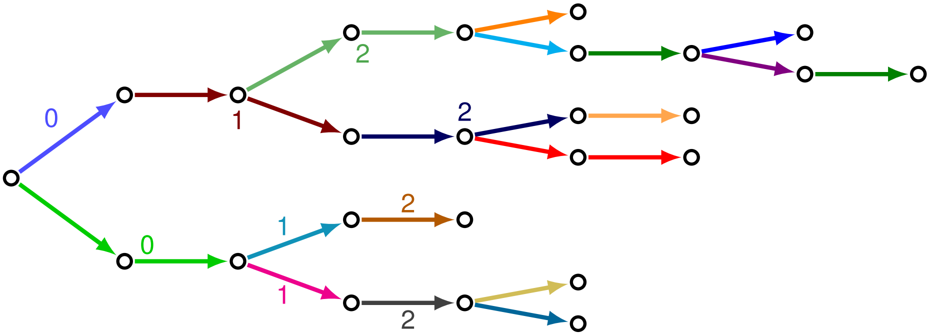
Selecting the uncovered up-links R



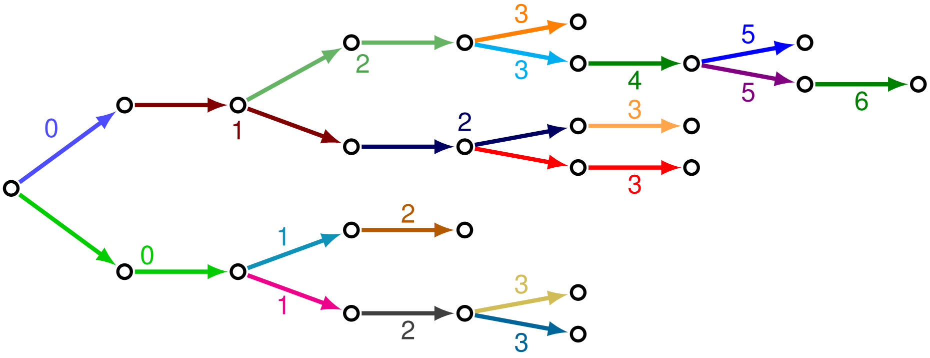
Selecting the uncovered up-links R



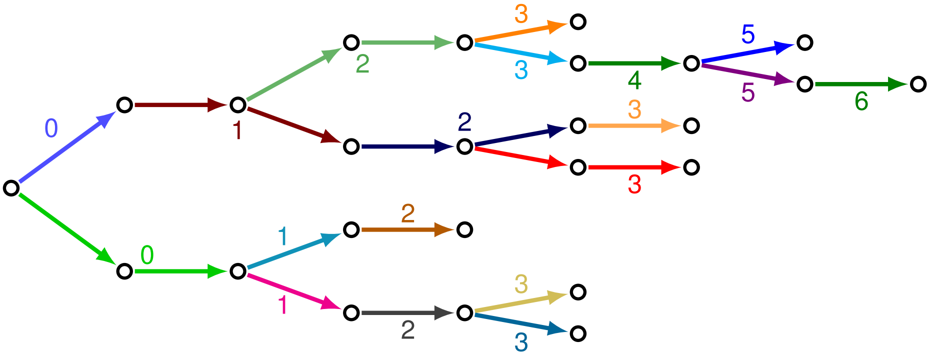
Selecting the uncovered up-links R



Selecting the uncovered up-links R



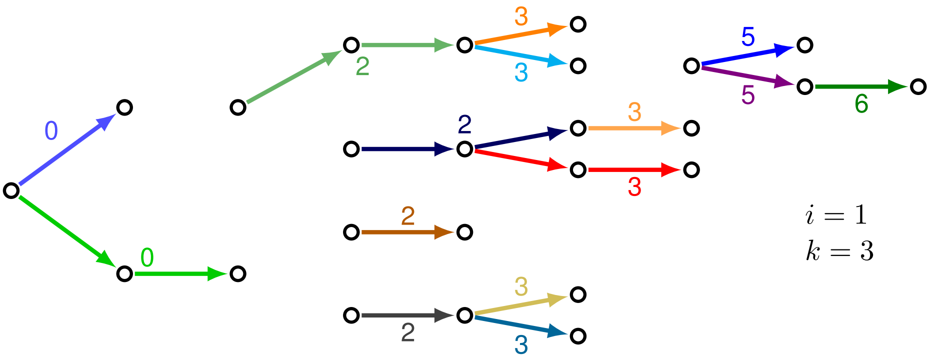
Selecting the uncovered up-links R



Sample $i \in \{0, \ldots, k-1\}$ uniformly at random.

$$R := \Big\{ u \in U \text{ : label of } A_u \text{ is in } \{\ i,\ i+k,\ i+2k,\ \ldots \} \Big\}$$

Selecting the uncovered up-links R



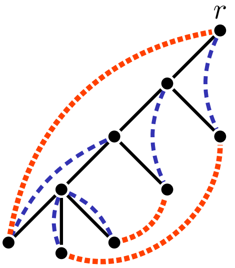
Sample $i \in \{0, \dots, k-1\}$ uniformly at random.

$$R := \Big\{ u \in U : \text{label of } A_u \text{ is in } \{\, i,\ i{+}k,\ i{+}2k,\ \dots \} \Big\}$$

The decomposition theorem

Fix $\varepsilon > 0$.

U := set of up-links s.t. the paths P_u with $u \in U$ are disjoint.



Decomposition Theorem

There exists a partition $\mathcal{C}$ of OPT into $\lceil 1/\varepsilon \rceil$-thin components s.t.:

$$\sum_{C \in \mathcal{C}} w(\mathrm{Drop}_U(C)) \geq (1-\varepsilon) \cdot w(U).$$

Conclusions

Conclusions

Summary

We gave two better-than-2 approximation algorithms for WTAP:

- Relative greedy: approximation ratio $1 + \ln 2 + \varepsilon \approx 1.69$
- Local search: approximation ratio $1.5 + \varepsilon$

Some open questions:

- Can we beat factor 2 for weighted connectivity augmentation?
- Can we beat factor 2 for the min. weight 2-edge-connected spanning subgraph problem?
- What about LP relaxations?
- What about node connectivity?

Conclusions

Summary

We gave two better-than-2 approximation algorithms for WTAP:

- Relative greedy: approximation ratio $1 + \ln 2 + \varepsilon \approx 1.69$
- Local search: approximation ratio $1.5 + \varepsilon$

Some open questions:

- Can we beat factor 2 for weighted connectivity augmentation?
- Can we beat factor 2 for the min. weight 2-edge-connected spanning subgraph problem?
- What about LP relaxations?
- What about node connectivity?

Thank you!