Forest Augmentation Problem

Semester project under the supervision of Ola Svensson

Igor Martayan

June 14, 2022

How to make a network both resilient and affordable?

How to make a network both resilient and affordable?

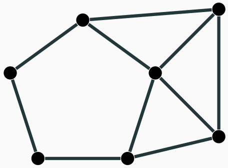
- resilient: stay connected even if some nodes / edges fail

How to make a network both resilient and affordable?

- resilient: stay connected even if some nodes / edges fail
- affordable: minimize the cost of the network

k-edge-connectivity

For every pair (s, t), there are at least k edge-disjoint paths between s and t

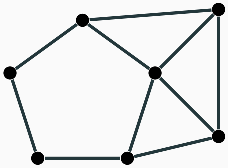


example of 2-edge-connected graph

k-edge-connectivity

For every pair (s, t), there are at least k edge-disjoint paths between s and t

most of these problems are NP-hard

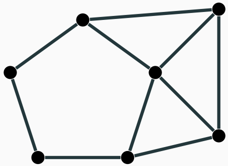


example of 2-edge-connected graph

k-edge-connectivity

For every pair (s, t), there are at least k edge-disjoint paths between s and t

most of these problems are NP-hard
$\rightarrow$ approximation algorithms



example of 2-edge-connected graph

Outline

- Graph augmentation: formulation, hardness, recent progress
- Matching augmentation: LP-based approximation algorithm

Graph augmentation problem

Graph augmentation

Input $G = (V, E)$, set of light edges $F \subseteq E$

Output Set of heavy edges $E' \subseteq E$ of min cardinality s.t. $(V, F \cup E')$ is 2-edge-connected

Graph augmentation

Input $G = (V, E)$, set of light edges $F \subseteq E$

Output Set of heavy edges $E' \subseteq E$ of min cardinality s.t. $(V, F \cup E')$ is 2-edge-connected

F can be a forest, a tree, a matching, a collection of paths...

Graph augmentation

Input $G = (V, E)$, set of light edges $F \subseteq E$

Output Set of heavy edges $E' \subseteq E$ of min cardinality s.t. $(V, F \cup E')$ is 2-edge-connected

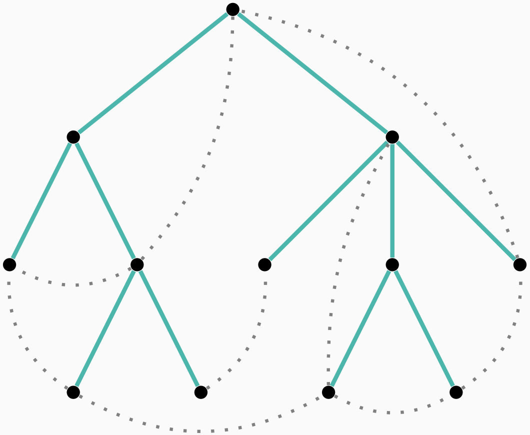
F can be a forest, a tree, a matching, a collection of paths...

Weighted version

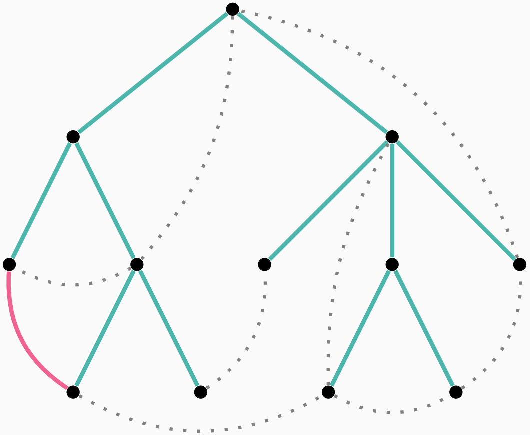
Input $G = (V, E)$, $w : E \mapsto \mathbb{R}^+$, set of light edges $F \subseteq E$

Output Set of heavy edges $E' \subseteq E$ of min weight s.t. $(V, F \cup E')$ is 2-edge-connected

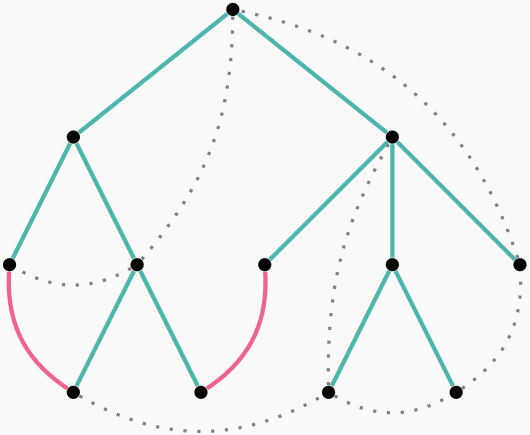
Example of graph augmentation



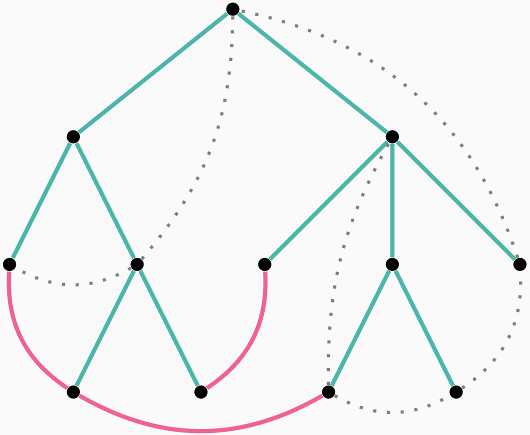
Example of graph augmentation



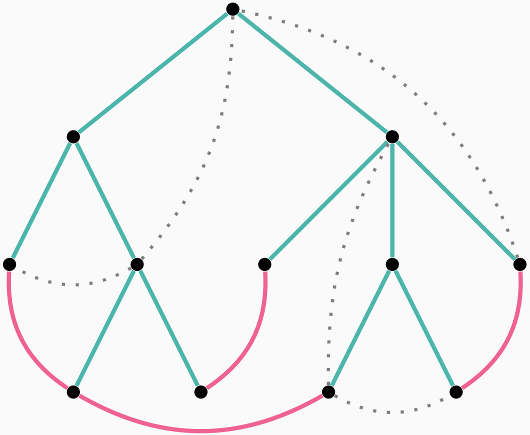
Example of graph augmentation



Example of graph augmentation



Example of graph augmentation



k-edge-connectivity, cut formulation

Every nontrivial cut has at least k edges

(special case of max-flow min-cut)

k-edge-connectivity, cut formulation

Every nontrivial cut has at least k edges

(special case of max-flow min-cut)

Linear Program

$$\min \sum_{e \in E \setminus F} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall \emptyset \subsetneq S \subsetneq V$$

$$x_e \in \{0, 1\} \quad \forall e \in E$$

x_e indicates whether e is selected

k-edge-connectivity, cut formulation

Every nontrivial cut has at least k edges

(special case of max-flow min-cut)

Relaxed Linear Program

$$\min \sum_{e \in E \setminus F} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall \varnothing \subsetneq S \subsetneq V$$

$$0 \leq x \leq 1 \quad \forall e \in E$$

x_e indicates whether e is selected

APX

NP optimization problems that have a poly-time approximation algorithm with a constant approximation ratio

APX

NP optimization problems that have a poly-time approximation algorithm with a constant approximation ratio

APX-hardness

every problem in APX can be reduced to this problem with an approximation-preserving poly-time reduction

APX

NP optimization problems that have a poly-time approximation algorithm with a constant approximation ratio

APX-hardness

every problem in APX can be reduced to this problem with an approximation-preserving poly-time reduction

Theorem (Kortsarz & al. 2004)

Tree Augmentation is APX-hard

$\exists\varepsilon$ s.t. approximating TAP with ratio $1+\varepsilon$ is NP-hard

Weighted Tree Augmentation

- [Traub & al. 2021] $1 + \ln 2 + \varepsilon$ (Relative Greedy)

Weighted Tree Augmentation

- [Traub & al. 2021] $1 + \ln 2 + \varepsilon$ (Relative Greedy)
- [Traub & al. 2022] $1.5 + \varepsilon$ (Local Search)

Weighted Tree Augmentation

- [Traub & al. 2021] $1 + \ln 2 + \varepsilon$ (Relative Greedy)
- [Traub & al. 2022] $1.5 + \varepsilon$ (Local Search)

Forest Augmentation
[Grandoni & al. 2022] < 2
(Reduction to path augmentation)

Approximating Matching Augmentation

[Bamas, Drygala, Svensson, 2022] *A Simple LP-Based Approximation Algorithm for the Matching Augmentation Problem*

[Bamas, Drygala, Svensson, 2022] *A Simple LP-Based Approximation Algorithm for the Matching Augmentation Problem*

- main result: better-than-2 approximation for MAP
- using only cut-LP and a DFS tree

$$\min \sum_{e \in E \setminus M} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall \varnothing \subsetneq S \subsetneq V$$

$$0 \leq x \leq 1 \quad \forall e \in E$$

Algorithm 1: LP-based approximation algorithm for MAP

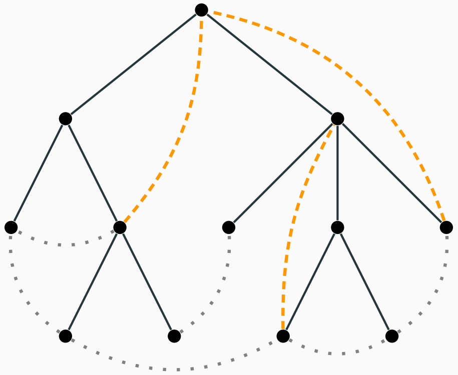
Input: A graph $G = (V, E)$ and a matching $M \subseteq E$.

$x^* \leftarrow$ optimal extreme point solution to $LP(G, M)$
$G' \leftarrow (V, \mathrm{Support}(x^*))$
$T \leftarrow$ LightDFS(G', M) // depth-first-search tree prioritizing light edges
$y^* \leftarrow$ optimal extreme point solution to $LP(G', T)$
$A \leftarrow \mathrm{Support}(y^*)$
return $T \cup A$

Why are we interested in a DFS tree?

DFS tree and up-links

- after building a DFS tree, the remaining edges are up-links
- up-link: one endpoint is an ancestor of the other



Having only up-links makes the problem tractable

Up-links and tractability

Having only up-links makes the problem tractable

Total unimodularity

For every square submatrix M of A, $\det M \in \{-1, 0, 1\}$

Having only up-links makes the problem tractable

Total unimodularity

For every square submatrix M of A, $\det M \in \{-1, 0, 1\}$

Theorem

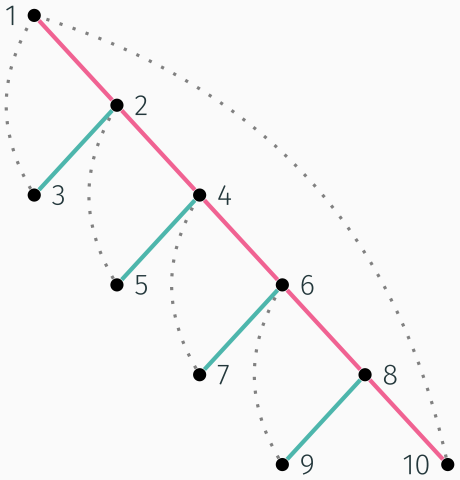
If $A \in \mathbb{R}^{m \times n}$ is totally unimodular and $b \in \mathbb{Z}^m$, then the extreme point solutions of

$$\begin{cases} Ax \geq b \\ x \in [0,1]^n \end{cases}$$

are integral

Why should we restrict the DFS tree to the support?

If we did not restrict the DFS tree to the support



Where does the better-than-2 ratio come from?

Half-integral edges

Suppose $\forall e, x_e^* \in \left\{0, \frac{1}{2}, 1\right\}$

Half-integral edges

Suppose $\forall e, x_e^* \in \left\{0, \frac{1}{2}, 1\right\}$

- cost of the DFS tree: $\mathrm{cost}(T) = n - 1 - |M|$

Half-integral edges

Suppose $\forall e, x_e^* \in \{0, \frac{1}{2}, 1\}$

- cost of the DFS tree: $\text{cost}(T) = n - 1 - |M|$
- "overcost" of T:

$$\text{cost}(T) - x^*(T \setminus M) = \sum_{e \in T \setminus M} \underbrace{1 - x_e^*}_{\leq \frac{1}{2}}$$
$$\leq \frac{n - 1 - |M|}{2}$$
$$\leq \frac{x^*(E \setminus M)}{2}$$

Half-integral edges

Suppose $\forall e, x_e^* \in \{0, \frac{1}{2}, 1\}$

- cost of the DFS tree: $\mathrm{cost}(T) = n - 1 - |M|$
- "overcost" of T:
$$\mathrm{cost}(T) - x^*(T \setminus M) \leq \frac{x^*(E \setminus M)}{2}$$
- since x^* is feasible for $LP(G', T)$,
$$y^*(E \setminus T) \leq x^*(E \setminus T)$$

Half-integral edges

Suppose $\forall e, x^*_e \in \{0, \frac{1}{2}, 1\}$

- cost of the DFS tree: $\text{cost}(T) = n - 1 - |M|$
- "overcost" of T:

$$\text{cost}(T) - x^*(T \setminus M) \leq \frac{x^*(E \setminus M)}{2}$$

- since x^* is feasible for $LP(G', T)$,

$$y^*(E \setminus T) \leq x^*(E \setminus T)$$

- total cost:

$$\text{cost}(T) + y^*(E \setminus T) \leq x^*(T \setminus M) + \frac{x^*(E \setminus M)}{2} + x^*(E \setminus T) = \frac{3}{2} x^*(E \setminus M)$$

Bounding the number of fractional edges

Each fractional edge is associated to a tight constraint

$$\mathcal{S} = \{S \subsetneq V; x^*(\delta(S)) = 2\}$$

Bounding the number of fractional edges

Each fractional edge is associated to a tight constraint

$$\mathcal{S} = \{S \subsetneq V; x^*(\delta(S)) = 2\}$$

Theorem

$\mathcal{S}$ can be reduced to a laminar family

Each fractional edge is associated to a tight constraint

$$\mathcal{S} = \{S \subsetneq V; x^*(\delta(S)) = 2\}$$

Theorem

$\mathcal{S}$ can be reduced to a laminar family

Laminar family

$\mathcal{F} \subseteq 2^V$ is laminar: for every sets $A, B \in \mathcal{F}$, either $A \subseteq B$, $B \subseteq A$ or $A \cap B = \varnothing$

Each fractional edge is associated to a tight constraint

$$\mathcal{S} = \{S \subsetneq V; x^*(\delta(S)) = 2\}$$

Theorem

$\mathcal{S}$ can be reduced to a laminar family

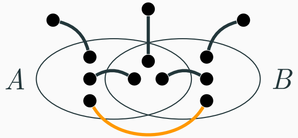
Laminar family

$\mathcal{F} \subseteq 2^V$ is laminar: for every sets $A, B \in \mathcal{F}$, either $A \subseteq B$, $B \subseteq A$ or $A \cap B = \varnothing$

Cardinality of a laminar family

If $|V| = n$ and $\mathcal{F}$ is laminar, then $|\mathcal{F}| \leq 2n - 1$

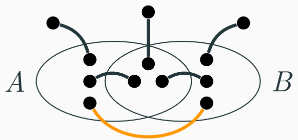
$$1_{\delta(A\cap B)} + 1_{\delta(A\cup B)} \leq 1_{\delta(A)} + 1_{\delta(B)}$$



Uncrossing property

If $S, T \in \mathcal{S}$ are not disjoint, then $S \cap T \in \mathcal{S}$ and $S \cup T \in \mathcal{S}$

$$1_{\delta(A\cap B)} + 1_{\delta(A\cup B)} \leq 1_{\delta(A)} + 1_{\delta(B)}$$



Uncrossing property

If $S, T \in \mathcal{S}$ are not disjoint, then $S \cap T \in \mathcal{S}$ and $S \cup T \in \mathcal{S}$

$$\operatorname{span}(\mathcal{F}) = \operatorname{span}\left\{1_{\delta(S)}; S \in \mathcal{F}\right\}$$

Theorem

Let $\mathcal{F}$ be the maximal laminar subfamily of $\mathcal{S}$, then $\operatorname{span}(\mathcal{F}) = \operatorname{span}(\mathcal{S})$

Conclusion and next directions

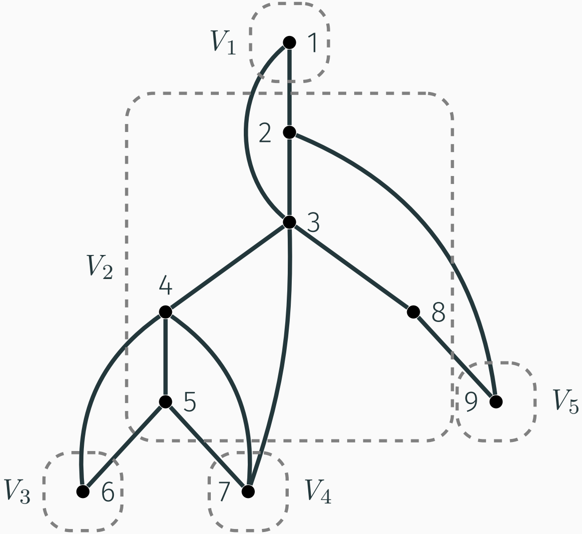
- forest augmentation is still a very active problem
- despite its hardness, the approximations keep improving
- analyzing it involves many elegant constructions

Next directions:

- refine the approximation ratio
- use other techniques such as tree carving
- extend to path augmentation

Appendix

Tree carving



DFS TREE WITH PATHS