Approximation Algorithms with Predictions

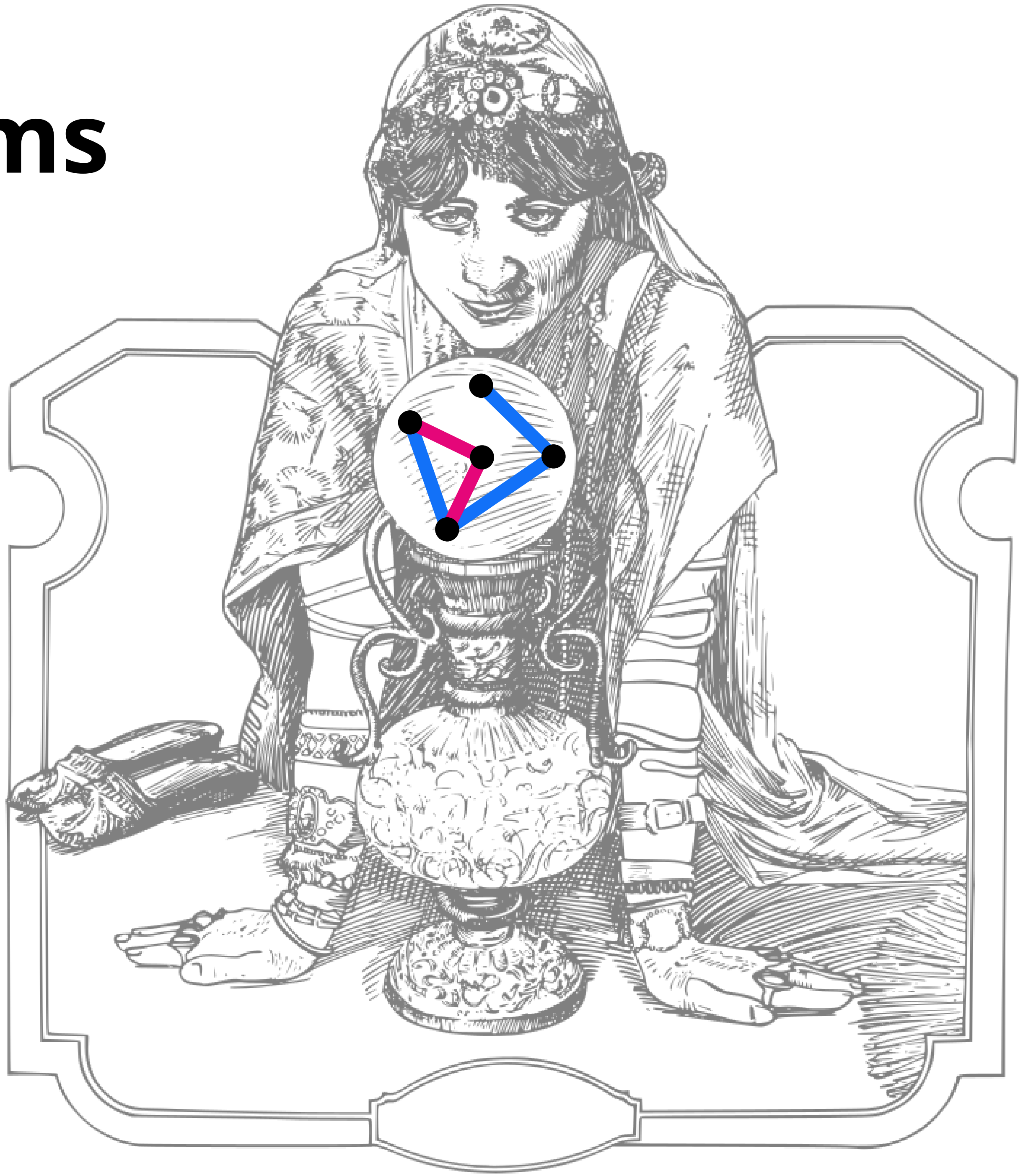
Antonios Antoniadis University of Twente

Marek Eliáš

Adam Polak

Moritz Venzin

Bocconi University



Approximation algorithms

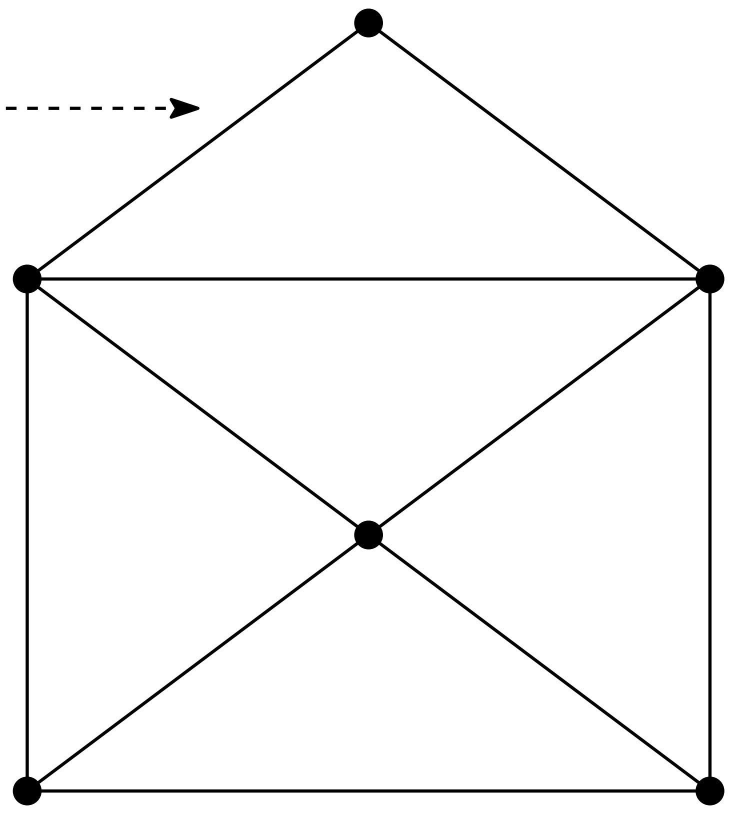
$$\text{value}(\text{ALG}) \leqslant \boldsymbol{\rho} \cdot \text{value}(\text{OPT})$$

approximation ratio (approximation factor)

Example: the Steiner Tree problem

Input:

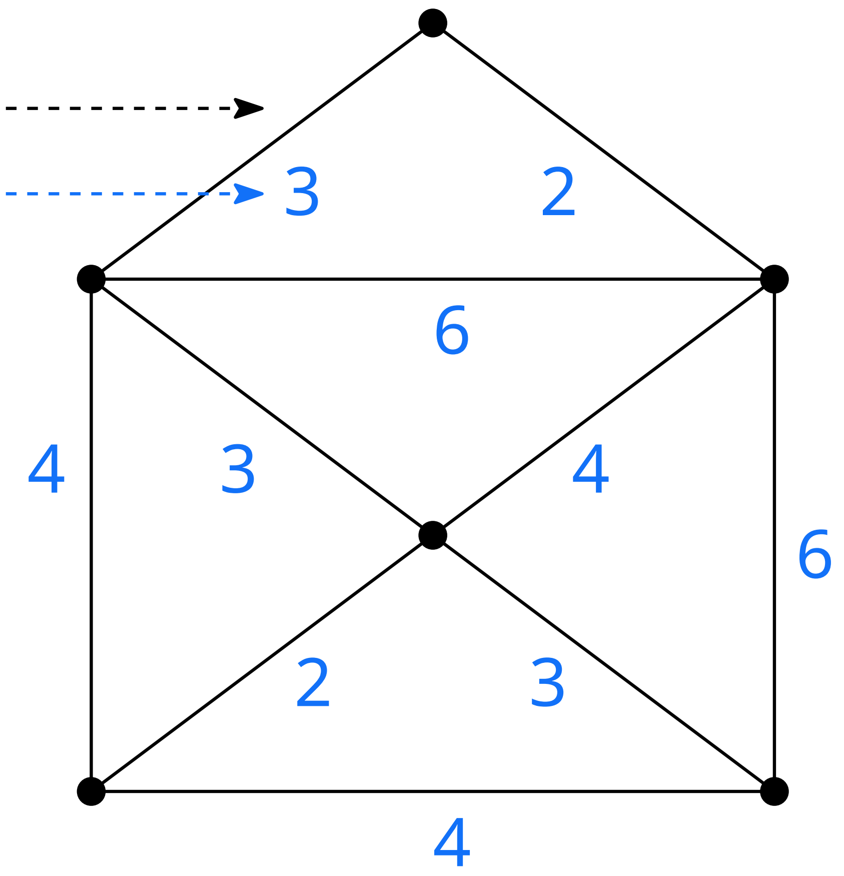
- undirected **graph** $G = (V, E)$



Example: the Steiner Tree problem

Input:

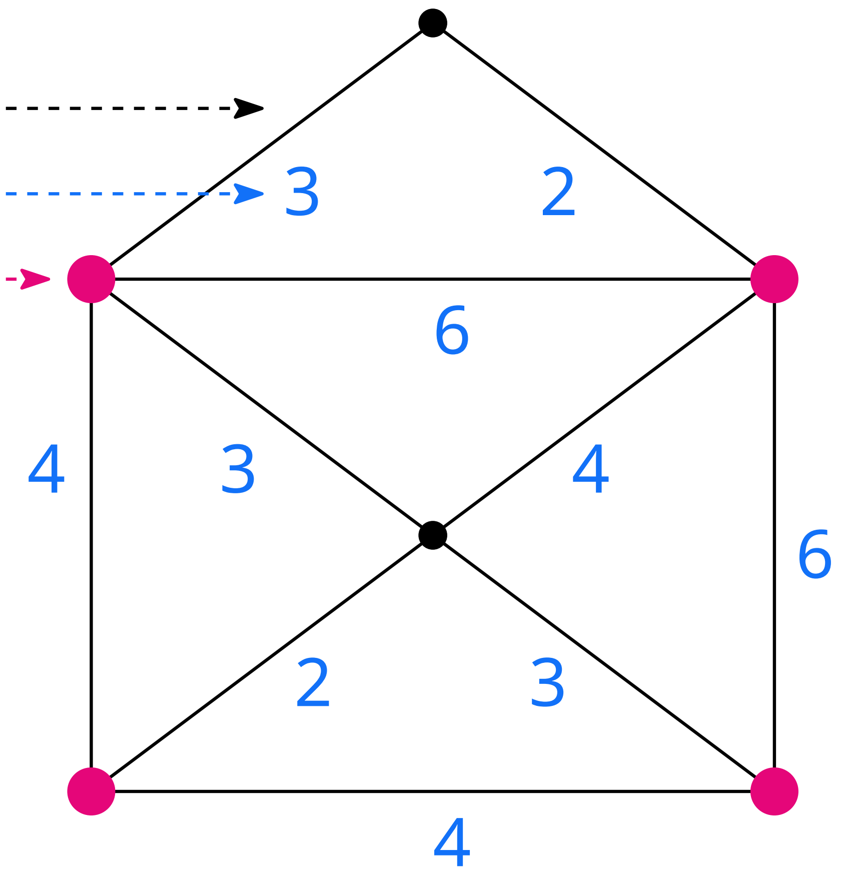
- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$



Example: the Steiner Tree problem

Input:

- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of **terminals** $T \subseteq V$



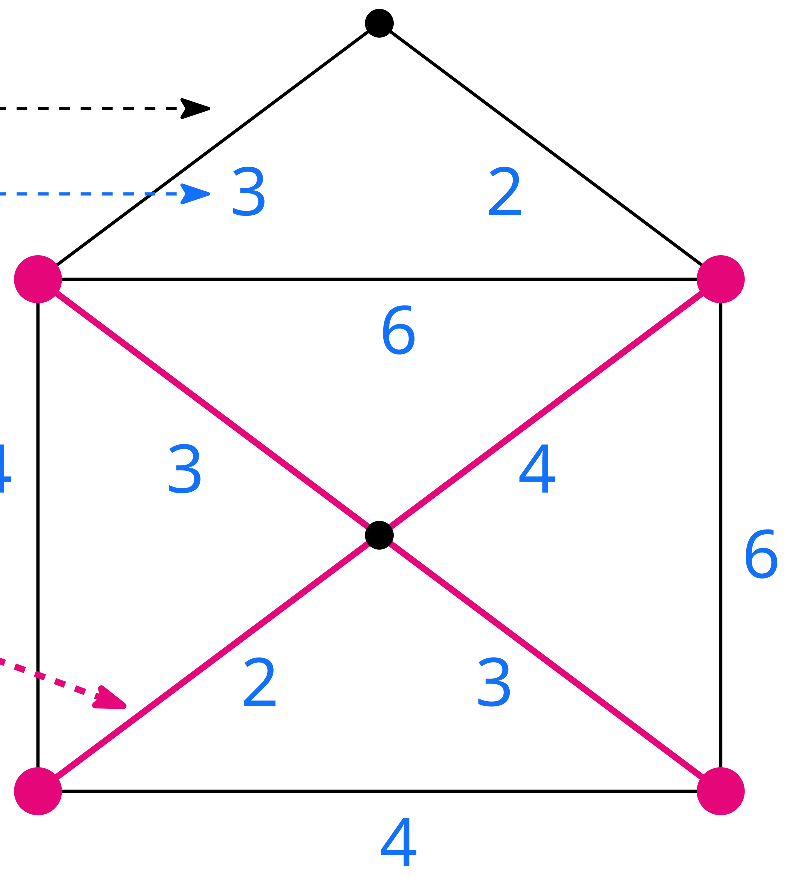
Example: the Steiner Tree problem

Input:

- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of **terminals** $T \subseteq V$

Output:

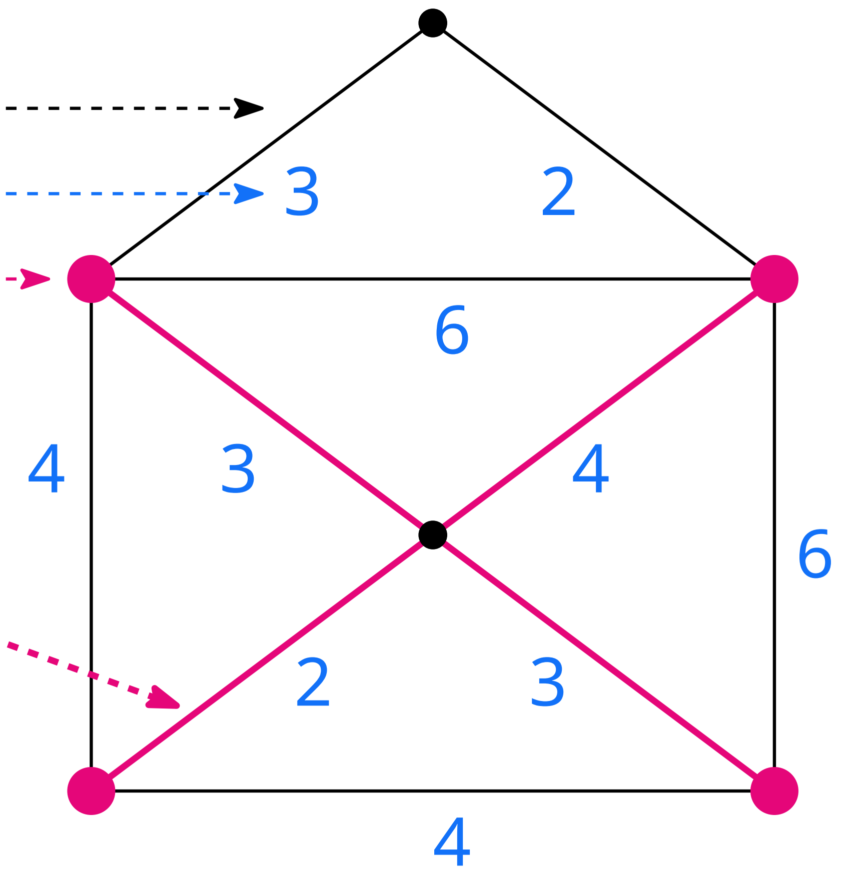
- min weight subgraph of G spanning T



Example: the Steiner Tree problem

Input:

- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of **terminals** $T \subseteq V$



Output:

- min weight subgraph of G spanning T

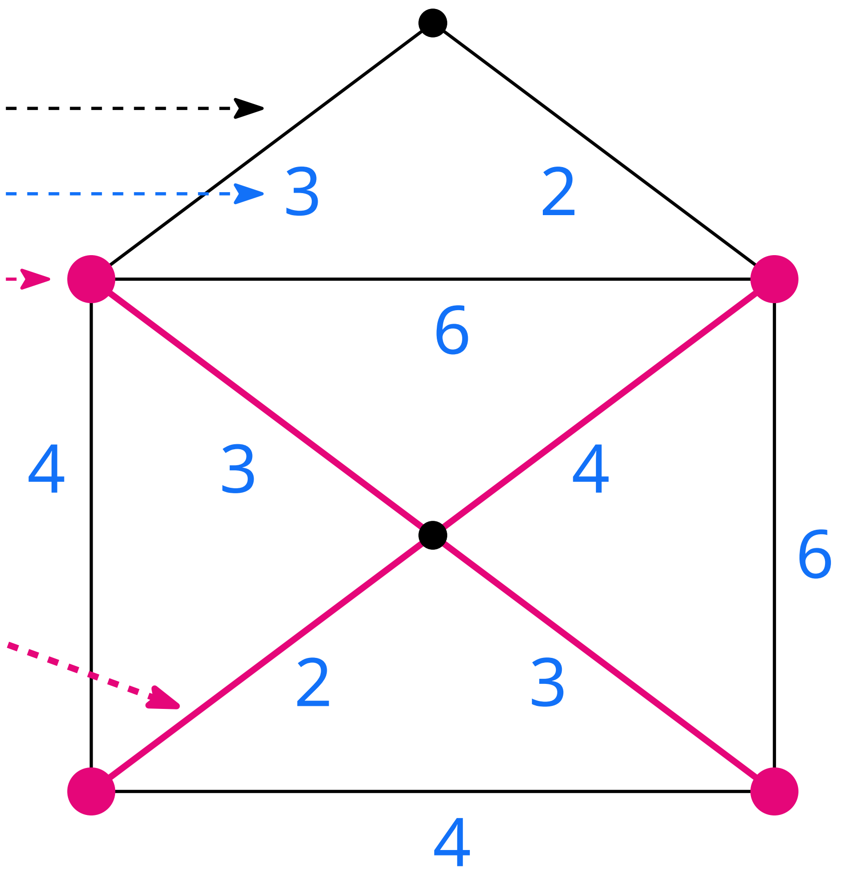
What is known?

- NP-hard
  [Karp ’72]

Example: the Steiner Tree problem

Input:

- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of **terminals** $T \subseteq V$



Output:

- min weight subgraph of G spanning T

What is known?

- NP-hard
  [Karp '72]
- **2**-approximation in (near-)**linear** $O(E + V \log V)$ time
  [Takahashi, Matsuyama '80], [Kou, Markowsky, Berman '81], [Wu, Widmayer, Wong '86], [Widmayer '86], [Mehlhorn '88]

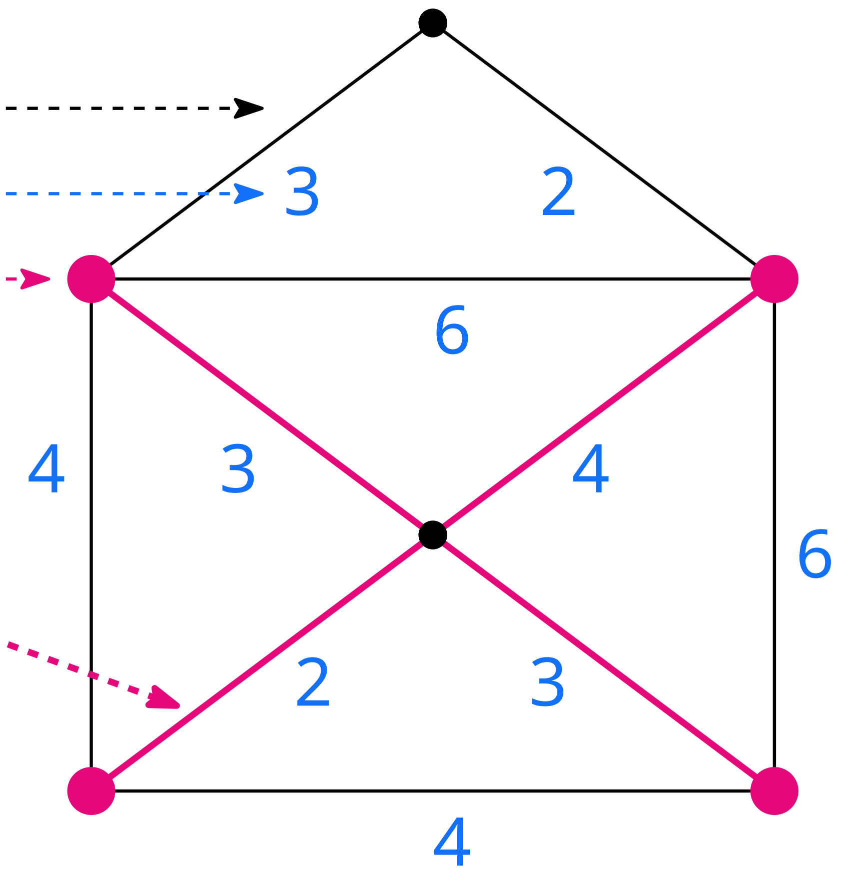
Example: the Steiner Tree problem

Input:

- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \to \mathbb{R}_{\geqslant 0}$
- set of **terminals** $T \subseteq V$

Output:

- min weight subgraph of G spanning T



What is known?

- NP-hard
  [Karp '72]
- **2**-approximation in (near-)**linear** $O(E + V \log V)$ time
  [Takahashi, Matsuyama '80], [Kou, Markowsky, Berman '81], [Wu, Widmayer, Wong '86], [Widmayer '86], [Mehlhorn '88]

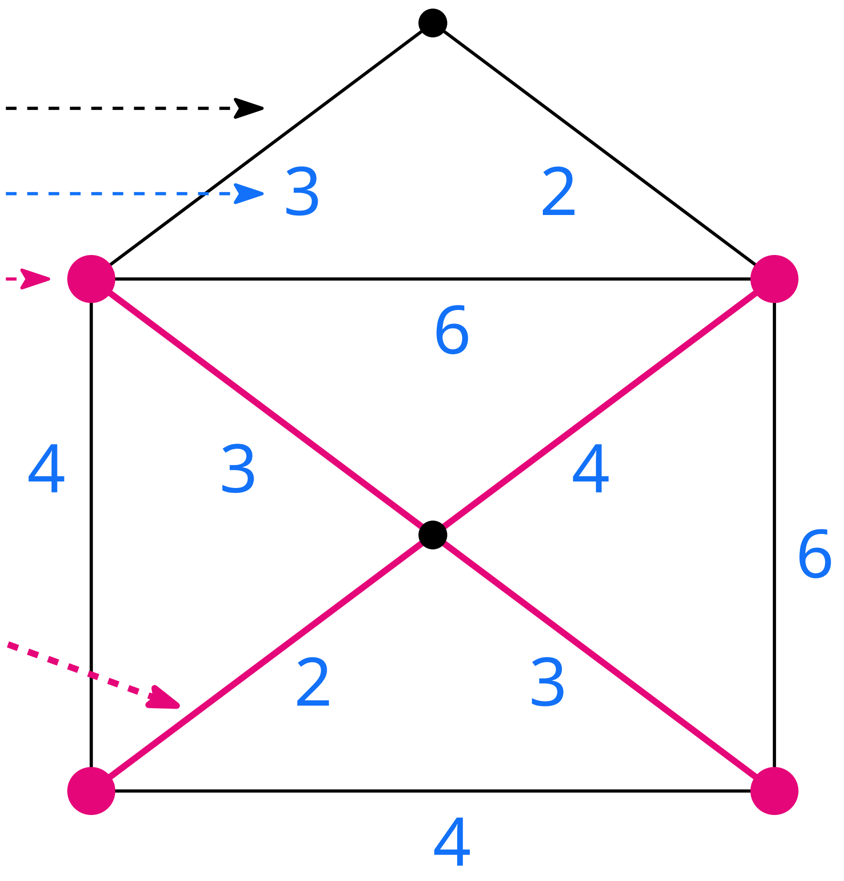
$$w(\mathrm{ALG}(I)) \leqslant \mathbf{2} \cdot w(\mathrm{OPT}(I))$$

for every instance I

Example: the Steiner Tree problem

Input:

- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of **terminals** $T \subseteq V$



Output:

- min weight subgraph of G spanning T

$$w(\mathrm{ALG}(I)) \leqslant \mathbf{2} \cdot w(\mathrm{OPT}(I))$$

for every instance I

What is known?

- NP-hard
  [Karp '72]
- **2**-approximation in (near-)**linear** $O(E + V \log V)$ time
  [Takahashi, Matsuyama '80], [Kou, Markowsky, Berman '81], [Wu, Widmayer, Wong '86], [Widmayer '86], [Mehlhorn '88]
- **1.39**-approximation in unspecified polynomial $\boldsymbol{V^{O(1)}}$ time
  [Zelikovsky '93], [Prömel, Steger '97], [Karpiński, Zelikovsky '97], [Hougardy, Prömel '99], [Robins, Zelikovsky '00], [Byrka, Grandoni, Rothvoss, Sanità '10]

The dilemma

- 2-approximation in (near-)linear $O(E + V \log V)$ time

fast & bad **vs** good & slow

- 1.39-approximation in unspecified polynomial $V^{O(1)}$ time

The dilemma

- 2-approximation in (near-)linear $O(E + V \log V)$ time

fast & bad **vs** good & slow

- 1.39-approximation in unspecified polynomial $V^{O(1)}$ time

Could we have both **fast** and **good**?

The dilemma

- 2-approximation in (near-)linear $O(E + V \log V)$ time

fast & bad **vs** good & slow

- 1.39-approximation in unspecified polynomial $V^{O(1)}$ time

Could we have both **fast** and **good**?

Yes!*

*If we have accurate enough **predictions**

The dilemma

- 2-approximation in (near-)linear $O(E + V \log V)$ time

fast & bad **vs** good & slow

- 1.39-approximation in unspecified polynomial $V^{O(1)}$ time

Could we have both **fast** and **good**?

Yes!*

*If we have accurate enough **predictions**

Steiner Tree with predictions

Input:

- undirected graph $G = (V, E)$
- edge weights $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of terminals $T \subseteq V$

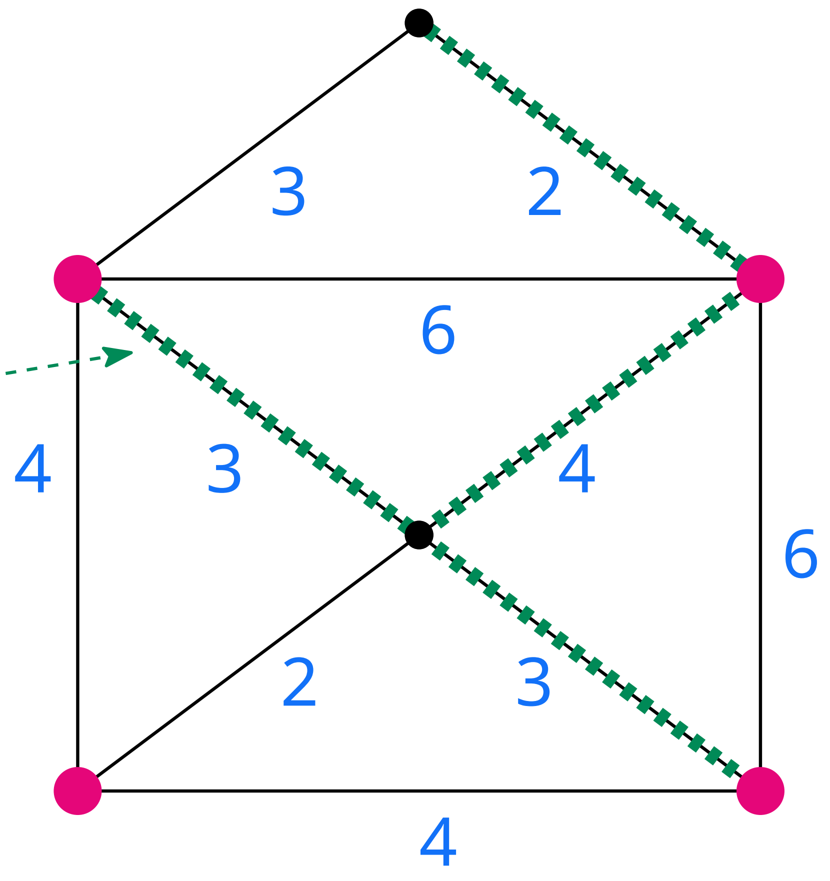
Prediction:

- **a subset of edges** $\mathrm{PRED} \subseteq E$

(not necessarily feasible)

Output:

- min weight subgraph of G spanning T



Steiner Tree with predictions

Input:

- undirected graph $G = (V, E)$
- edge weights $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of terminals $T \subseteq V$

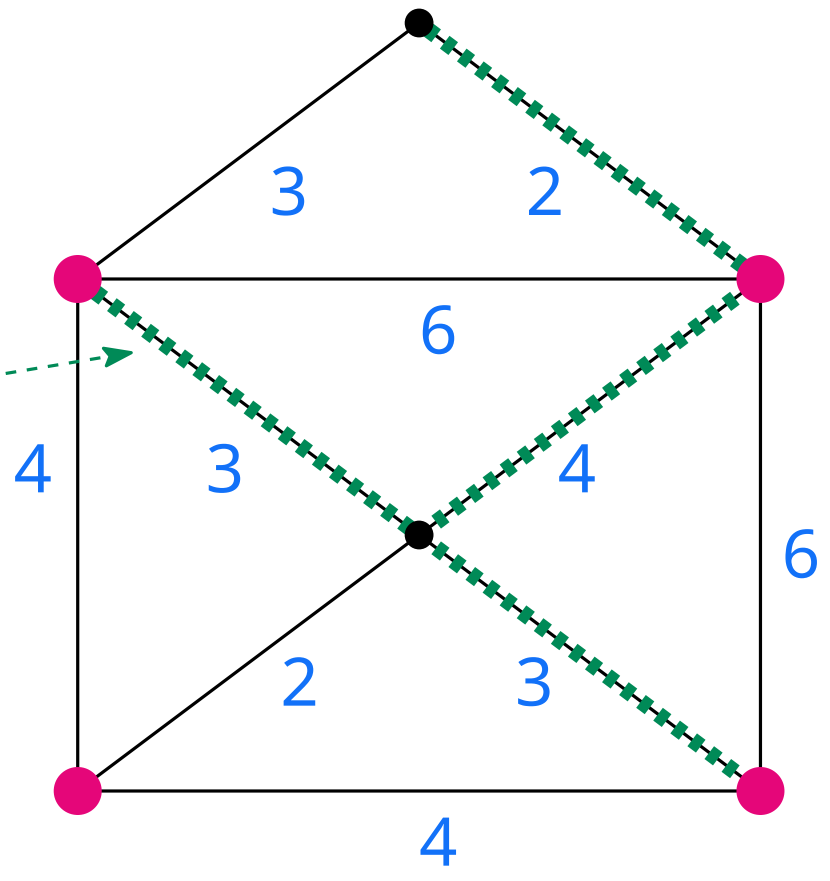
Prediction:

- **a subset of edges** $\mathrm{PRED} \subseteq E$

(not necessarily feasible)

Output:

- min weight subgraph of G spanning T



Our result:

- $(1 + \eta/\textsc{opt})$-approximation in (near-)linear $O(E + V \log V)$ time

prediction error $\boldsymbol{\eta}$ $:= w(\mathrm{PRED} \setminus \mathrm{OPT}) + w(\mathrm{OPT} \setminus \mathrm{PRED})$

Steiner Tree with predictions

Input:

- undirected graph $G = (V, E)$
- edge weights $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$
- set of terminals $T \subseteq V$

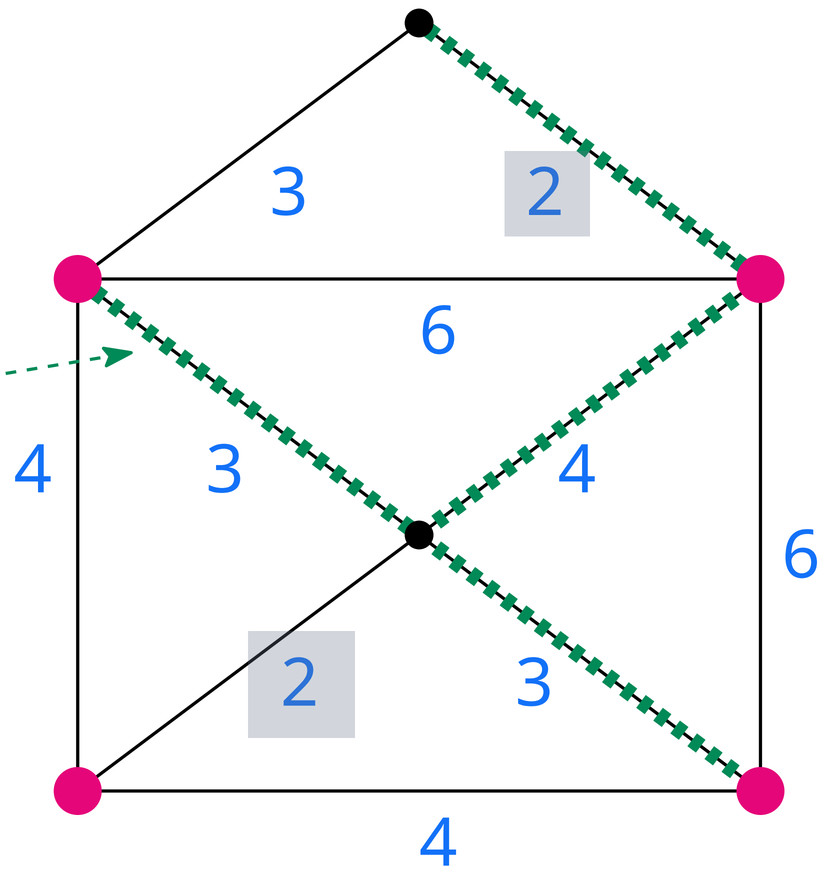
Prediction:

- **a subset of edges** PRED $\subseteq E$

(not necessarily feasible)

Output:

- min weight subgraph of G spanning T



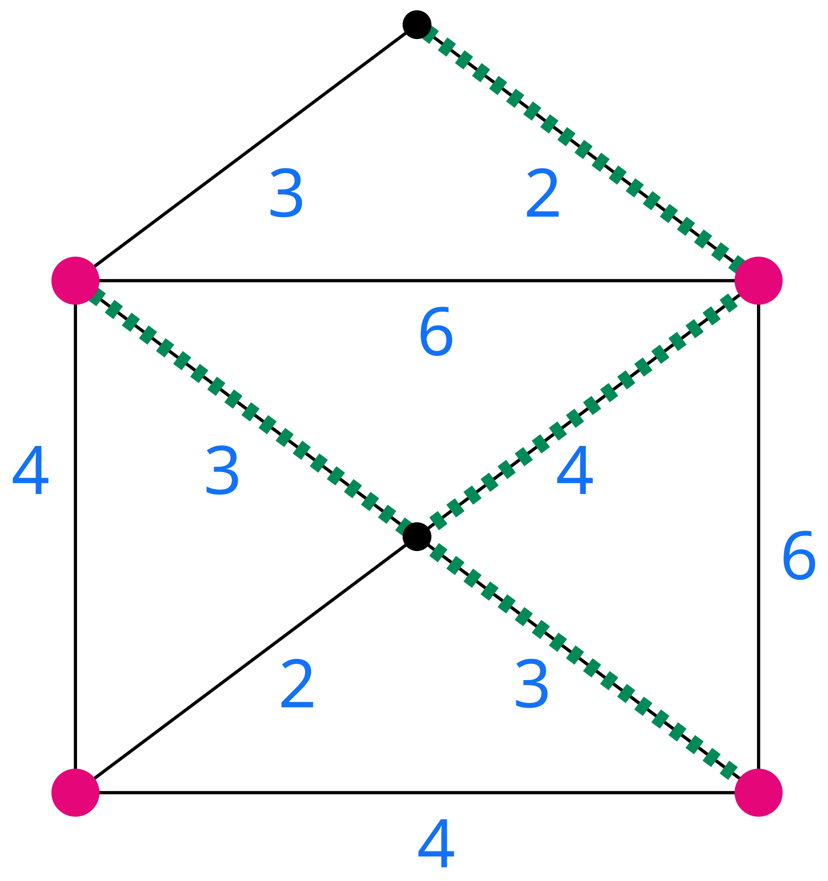
Our result:

- $(1 + \eta/\textsc{opt})$-approximation in (near-)linear $O(E + V \log V)$ time

prediction error η $:= w(\text{PRED} \setminus \text{OPT}) + w(\text{OPT} \setminus \text{PRED})$

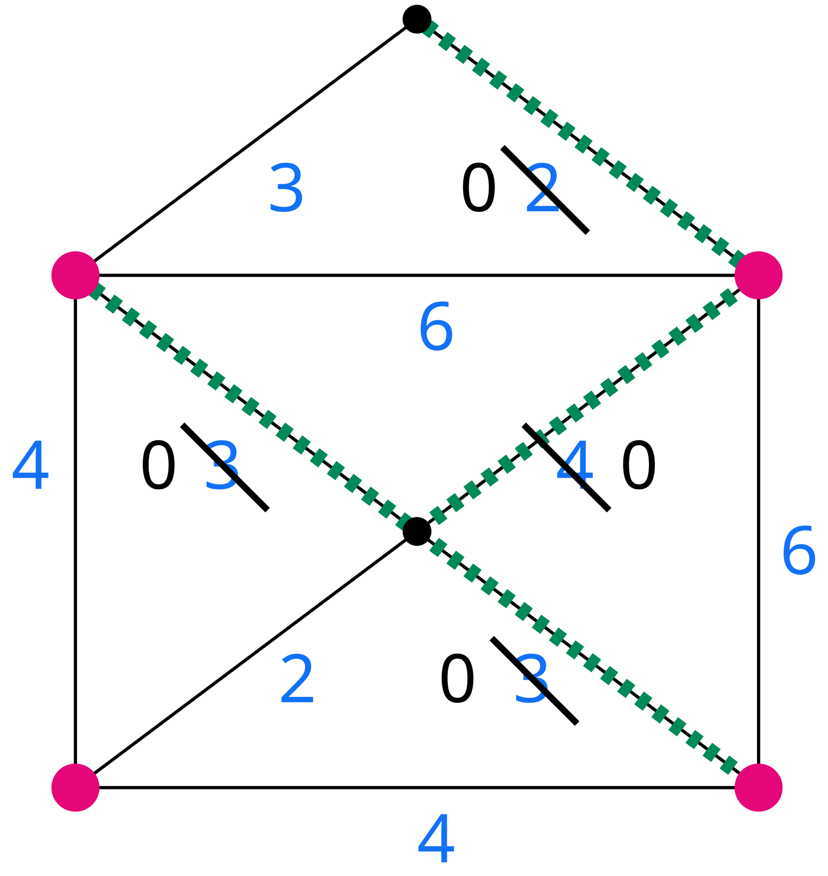
Our learning-augmented algorithm for Steiner Tree

- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$



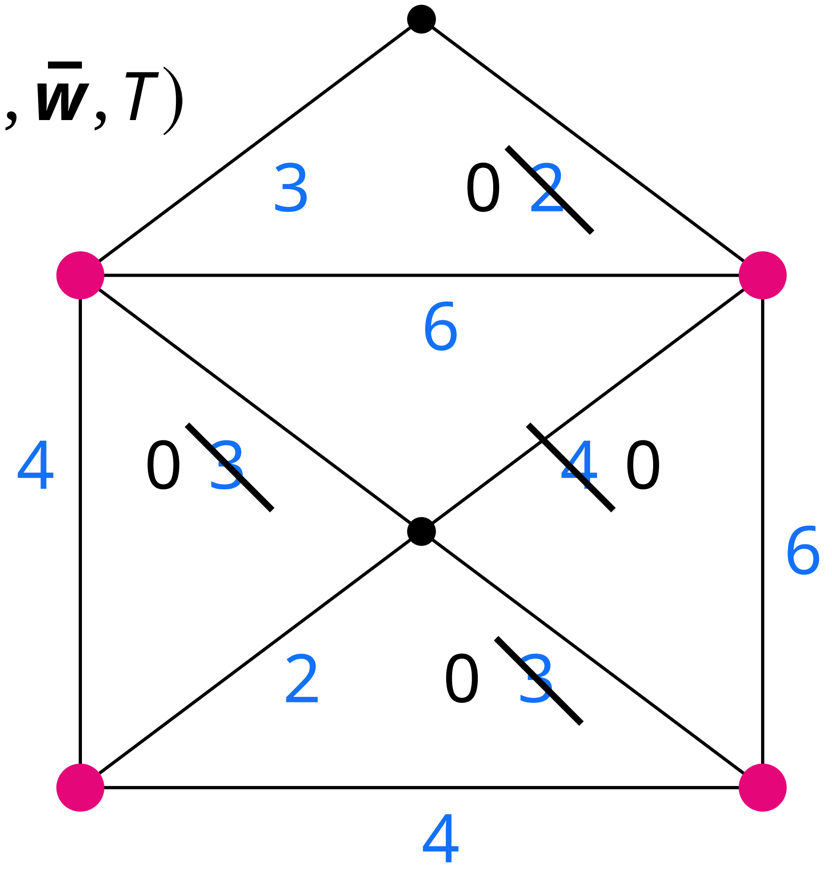
Our learning-augmented algorithm for Steiner Tree

- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$



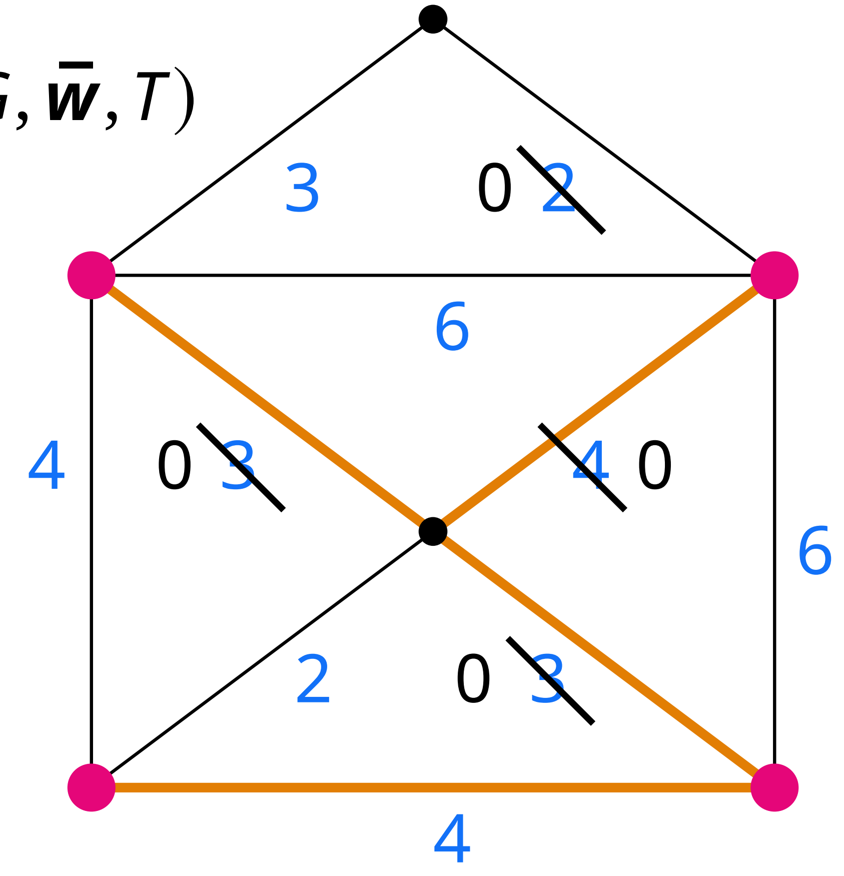
Our learning-augmented algorithm for Steiner Tree

- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $$\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$



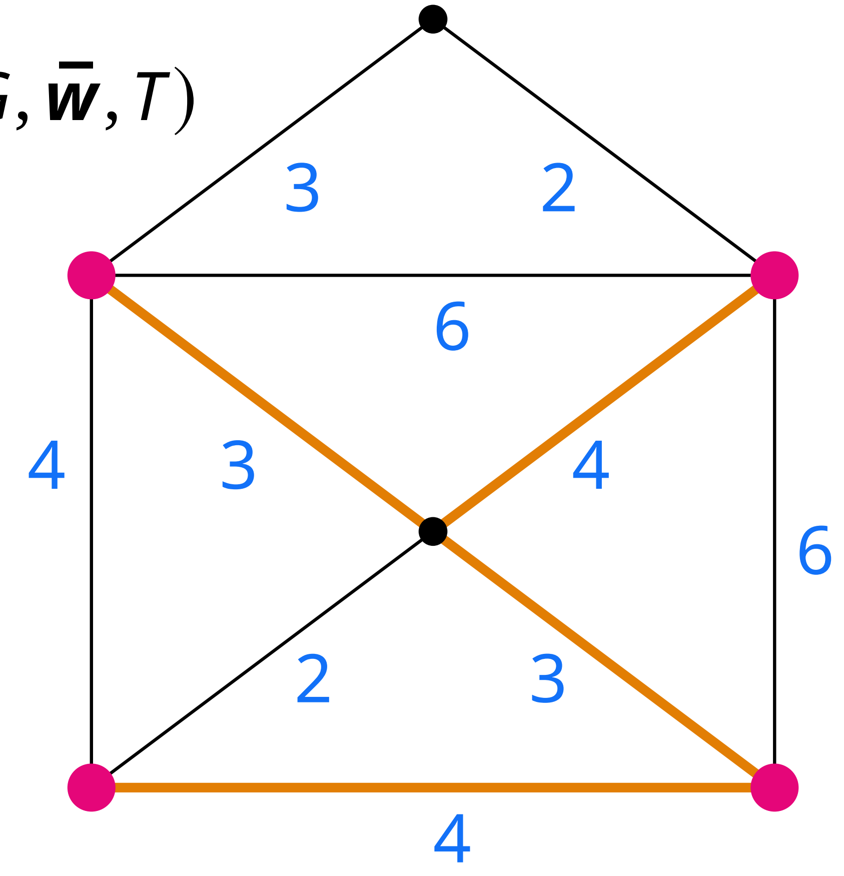
Our learning-augmented algorithm for Steiner Tree

- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\quad \bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$



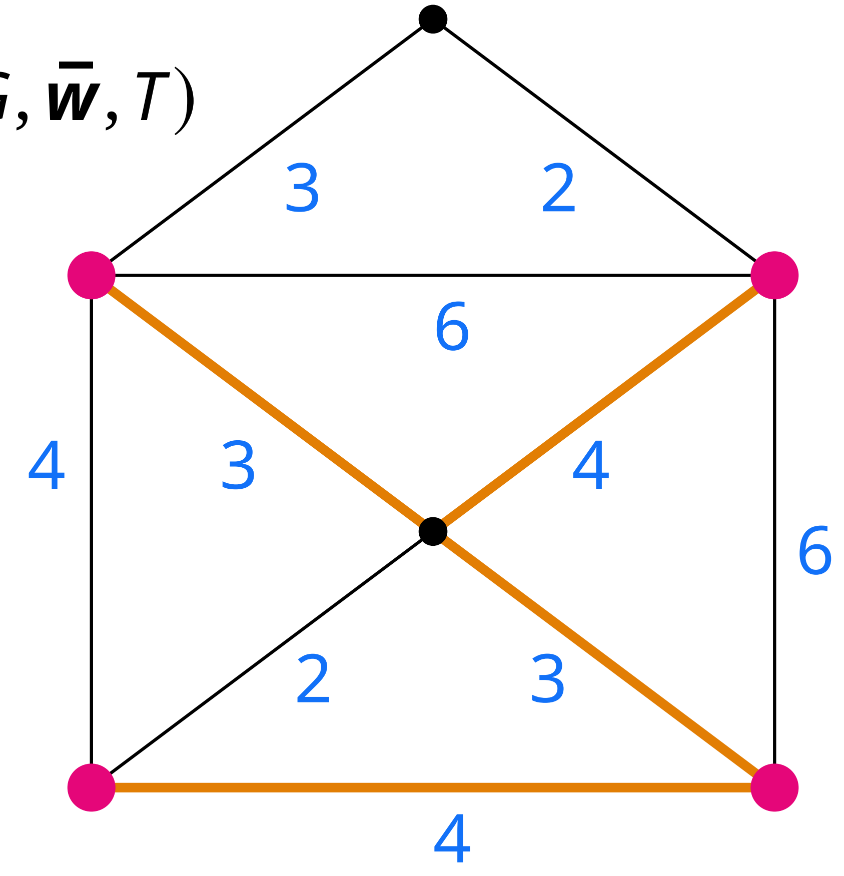
Our learning-augmented algorithm for Steiner Tree

- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\quad \bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned



Our learning-augmented algorithm for Steiner Tree

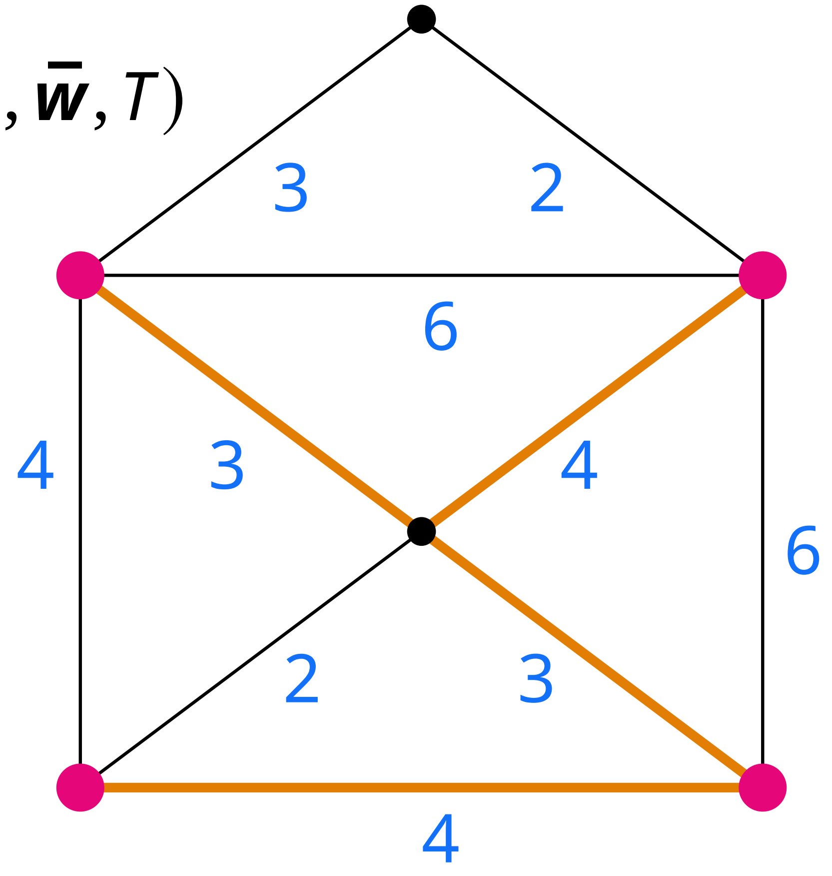
- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\quad \bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned



Claim: $\quad w(\mathrm{ALG}) \leqslant (1 + \eta/\mathrm{OPT}) \cdot \mathrm{OPT} = \mathrm{OPT} + \eta$

Our learning-augmented algorithm for Steiner Tree

- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \to \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned



Claim: $w(\mathrm{ALG}) \leqslant (1 + \eta/\mathrm{OPT}) \cdot \mathrm{OPT} = \mathrm{OPT} + \eta$

Proof: via simple analysis of a Venn diagram

Generalization

For any **minimization problem** of the following form:

Input:
- n items with **weights**: $w_1, w_2, \ldots, w_n \in \mathbb{R}_{\geqslant 0}$
- implicitly given set of **feasible solutions**: $\mathcal{X} \subseteq \{1, 2, \ldots, n\}$

Output:
- $\mathbf{min}\{\boldsymbol{w}(X) \mid X \in \mathcal{X}\}$

Generalization

For any **minimization problem** of the following form:

Input:

- n items with **weights**: $w_1, w_2, \ldots, w_n \in \mathbb{R}_{\geq 0}$
- implicitly given set of **feasible solutions**: $\mathcal{X} \subseteq \{1, 2, \ldots, n\}$

Output:

- $\mathbf{min}\{\boldsymbol{w}(X) \mid X \in \mathcal{X}\}$

if there is a $\boldsymbol{T(n)}$-time approximation algorithm with **approximation factor 2**

Generalization

For any **minimization problem** of the following form:

Input:
- n items with **weights**: $w_1, w_2, \ldots, w_n \in \mathbb{R}_{\geqslant 0}$
- implicitly given set of **feasible solutions**: $\mathcal{X} \subseteq \{1, 2, \ldots, n\}$

Output:
- $\mathbf{min}\{\boldsymbol{w}(X) \mid X \in \mathcal{X}\}$

if there is a $\boldsymbol{T(n)}$-time approximation algorithm with **approximation factor 2**

then there is an $\boldsymbol{O(T(n))}$-time **learning-augmented** approximation algorithm with **approximation factor** $\mathbf{1 + \frac{\eta}{OPT}}$

Generalization

For any **minimization problem** of the following form:

Input:

- n items with **weights**: $w_1, w_2, \ldots, w_n \in \mathbb{R}_{\geqslant 0}$
- implicitly given set of **feasible solutions**: $\mathcal{X} \subseteq \{1, 2, \ldots, n\}$

Output:

- $\mathbf{min}\{\boldsymbol{w}(X) \mid X \in \mathcal{X}\}$

if there is a $\boldsymbol{T(n)}$-time approximation algorithm with **approximation factor 2**

then there is an $\boldsymbol{O(T(n))}$-time **learning-augmented** approximation algorithm with **approximation factor** $\min\{2, \mathbf{1} + \frac{\boldsymbol{\eta}}{\mathbf{OPT}}\}$

Generalization

For any **minimization problem** of the following form:

Input:
- n items with **weights**: $w_1, w_2, \ldots, w_n \in \mathbb{R}_{\geqslant 0}$
- implicitly given set of **feasible solutions**: $\mathcal{X} \subseteq \{1, 2, \ldots, n\}$

Output:
- $\mathbf{min}\{\boldsymbol{w}(X) \mid X \in \mathcal{X}\}$

if there is a $\boldsymbol{T(n)}$-time approximation algorithm with **approximation factor** $\boldsymbol{\rho}$

then there is an $\boldsymbol{O(T(n))}$-time **learning-augmented** approximation algorithm with **approximation factor** $\min\{\rho, 1 + \frac{\eta_+ + (\rho - 1)\cdot\eta_-}{\mathrm{OPT}}\}$

$\eta_- := w(\mathrm{OPT} \setminus \mathrm{PRED})$

$\eta_+ := w(\mathrm{PRED} \setminus \mathrm{OPT})$

Applications

- (Minimum Weight) Steiner Tree
- (Minimum Weight) Vertex Cover
- Minimum Weight Perfect Matching in Metric Graphs
- (Maximum Weight) Clique
- Knapsack

(a similar general theorem for **maximization** problems)

- [place for your favorite problem]

What else is in the paper?

- **Lower bounds** showing optimality of the general framework
- **Refined** algorithm with stronger guarantees for **Steiner Tree**
- **Experimental evaluation**

What else is in the paper?

- **Lower bounds** showing optimality of the general framework
- **Refined** algorithm with stronger guarantees for **Steiner Tree**
- **Experimental evaluation**

Thank you!