Approximation Algorithms with Predictions

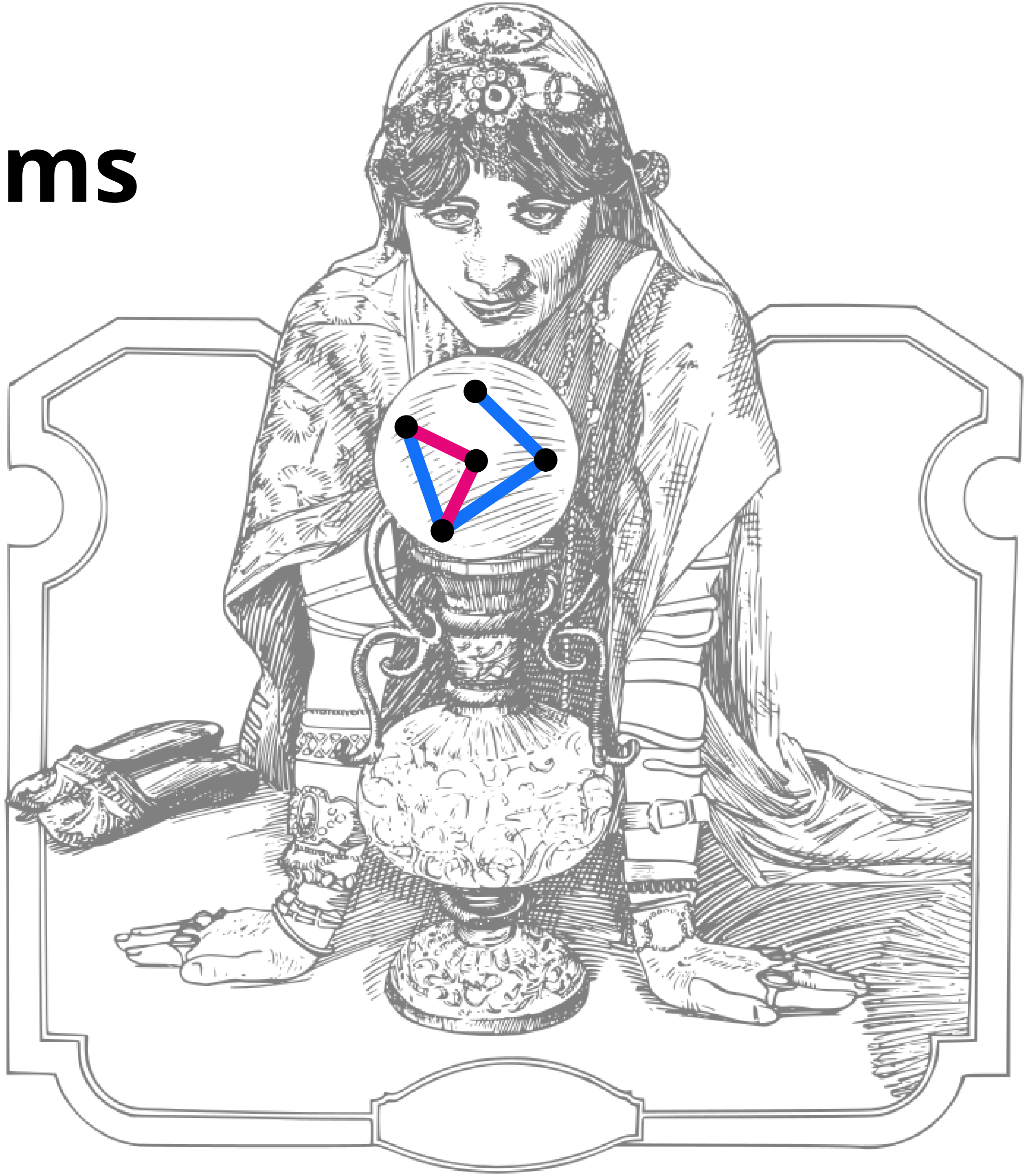
Antonios Antoniadis University of Twente

Marek Eliáš

Adam Polak

Moritz Venzin

Bocconi University



Classical algorithms

- worst-case guarantees
- overly pessimistic

Classical algorithms

- worst-case guarantees
- overly pessimistic

Machine learning

- powerful for typical inputs
- no guarantees, can go crazy

Classical algorithms

- worst-case guarantees
- overly pessimistic

Machine learning

- powerful for typical inputs
- no guarantees, can go crazy

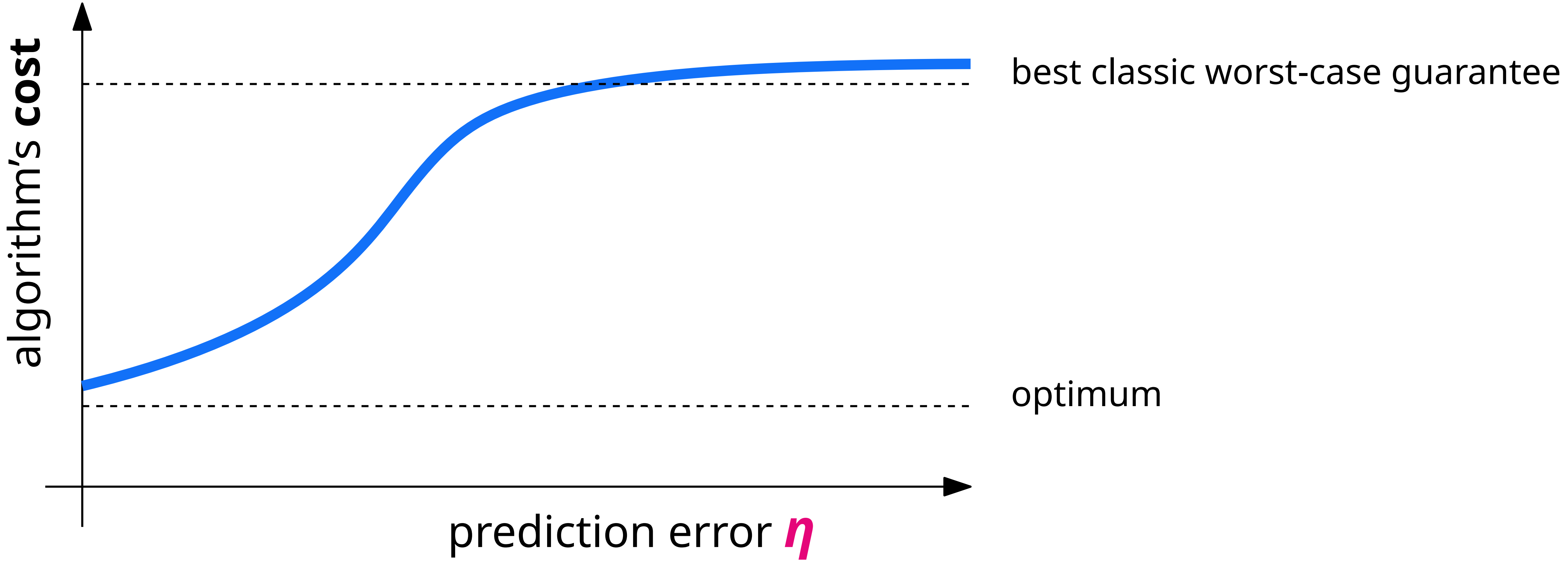
Best of both worlds: **algorithms with predictions**

Algorithms with predictions (a.k.a. learning-augmented algorithms)

Input + black-box **predictions** (possibly inacurate, with some **error** η)

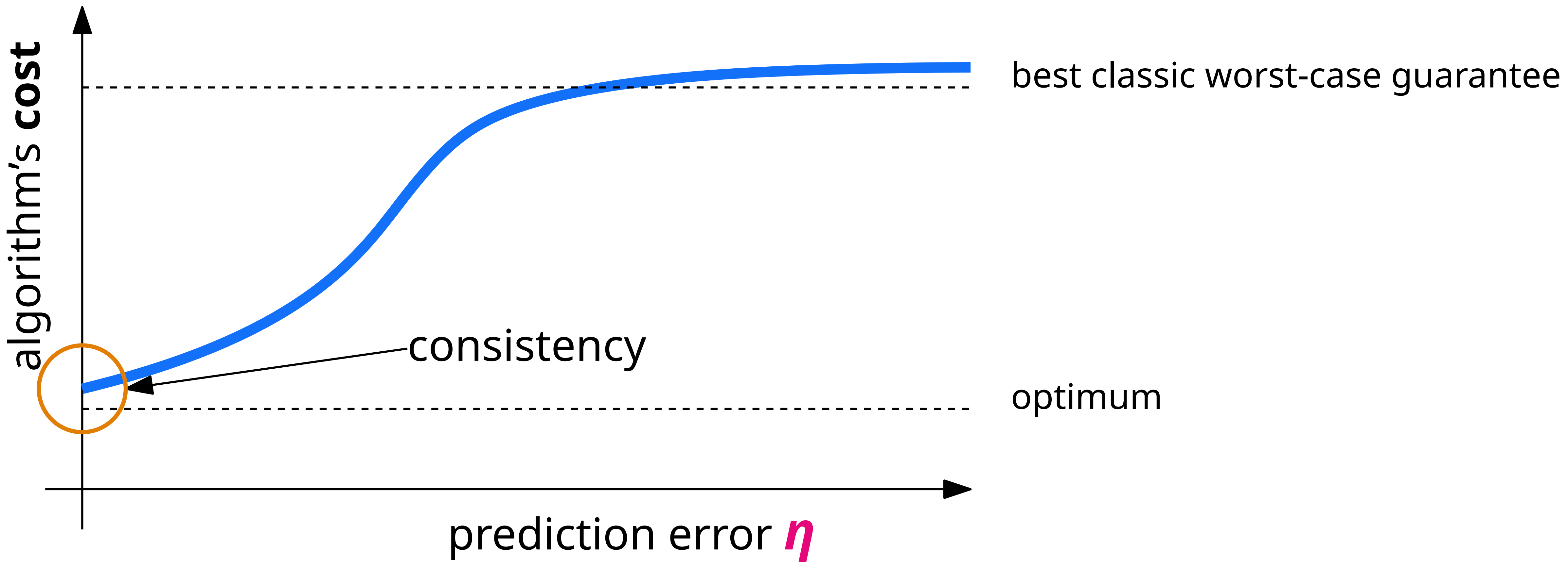
Algorithms with predictions (a.k.a. learning-augmented algorithms)

Input + black-box **predictions** (possibly inacurate, with some **error** η)



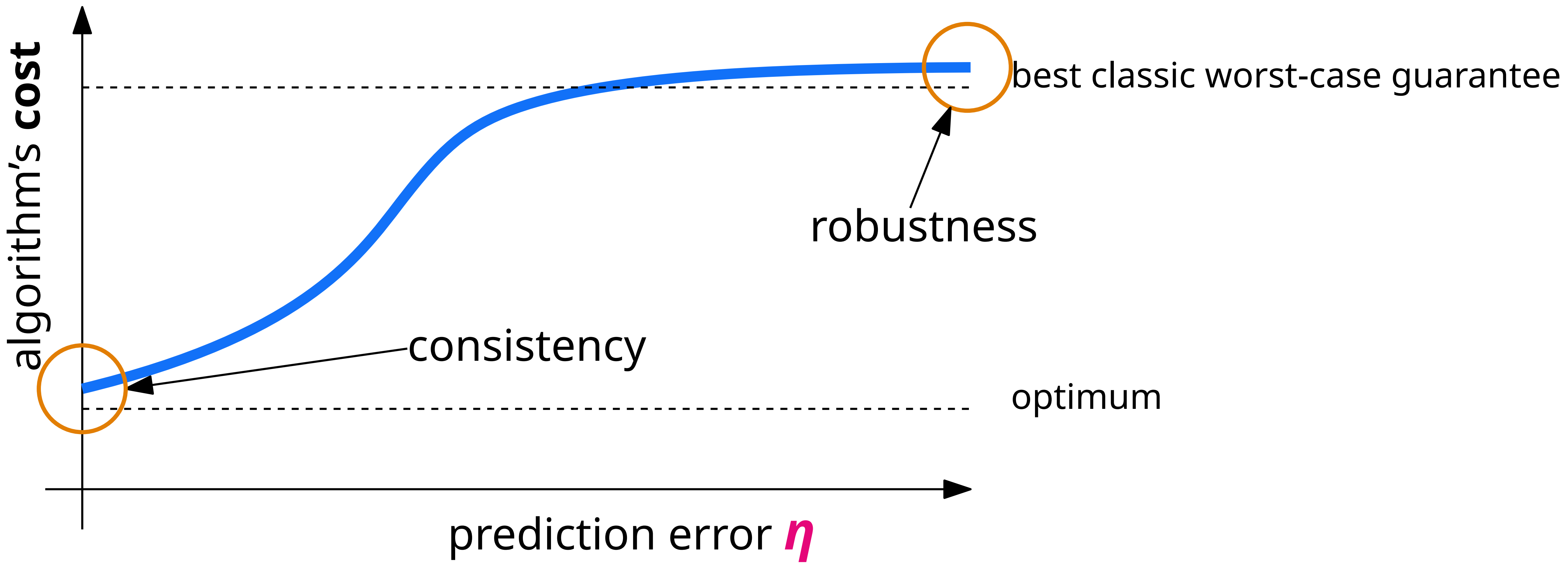
Algorithms with predictions (a.k.a. learning-augmented algorithms)

Input + black-box **predictions** (possibly inacurate, with some **error** η)



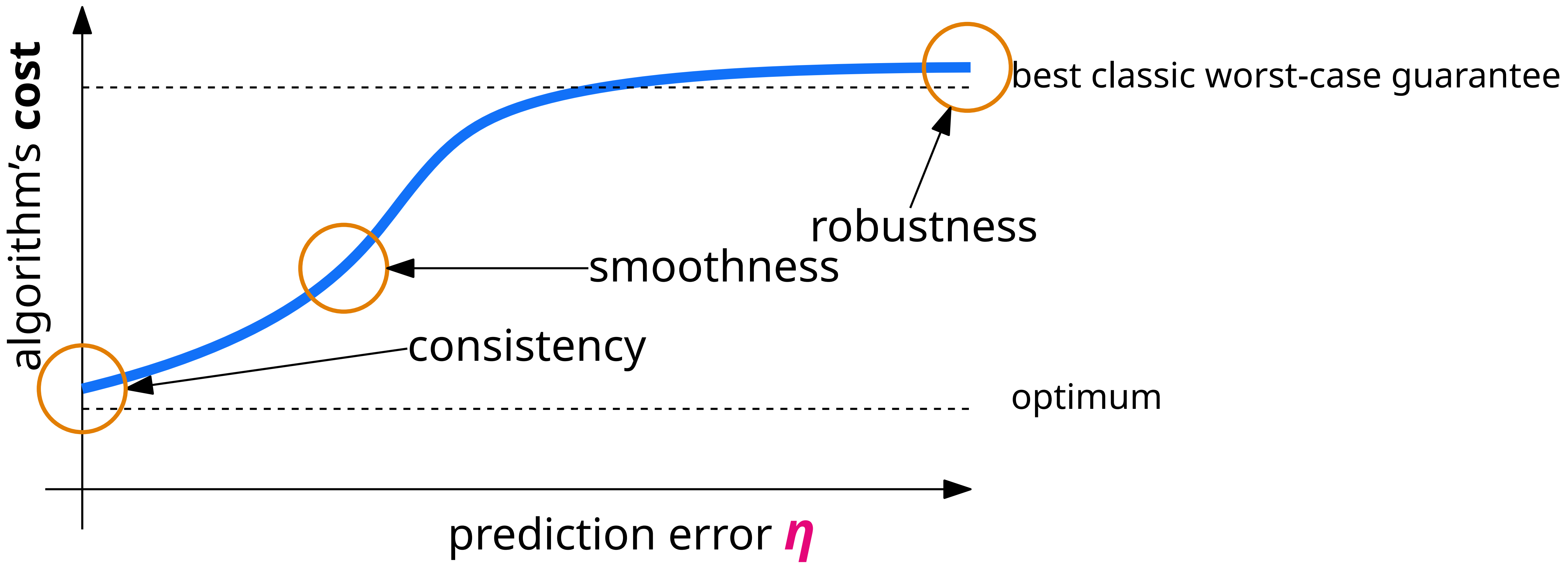
Algorithms with predictions (a.k.a. learning-augmented algorithms)

Input + black-box **predictions** (possibly inacurate, with some **error** η)



Algorithms with predictions (a.k.a. learning-augmented algorithms)

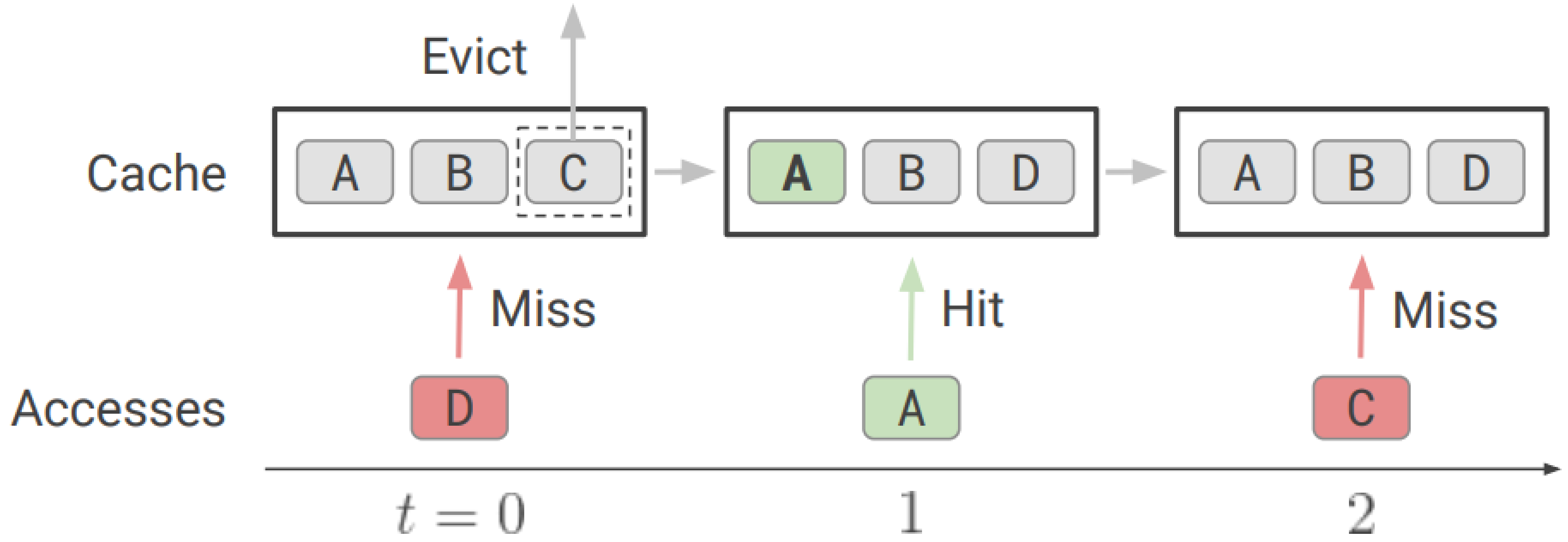
Input + black-box **predictions** (possibly inacurate, with some **error** η)



Predictions can improve **competitive ratio** of **online** algorithms

E.g.: **caching**

[Lykouris, Vassilvitskii, ICML'18]



source: arxiv.org/abs/2006.16239

Prediction: *When currently requested item will be requested again?*

Result: $\boldsymbol{O(\min(\log k, \sqrt{\eta / OPT}))}$-competitive algorithm   ($\Theta(\log k)$ is tight for classic)

Predictions can improve **running time** of **static** algorithms

E.g.: **max weight bipartite matching** [Dinitz, Im, Lavastida, Moseley, Vassilvitskii, NeurIPS'21]

Primal:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in N(v)} x_e = 1 \quad \forall\, v \in V \\ & x_e \geqslant 0 \quad \forall\, e \in E \end{aligned}$$

Dual:

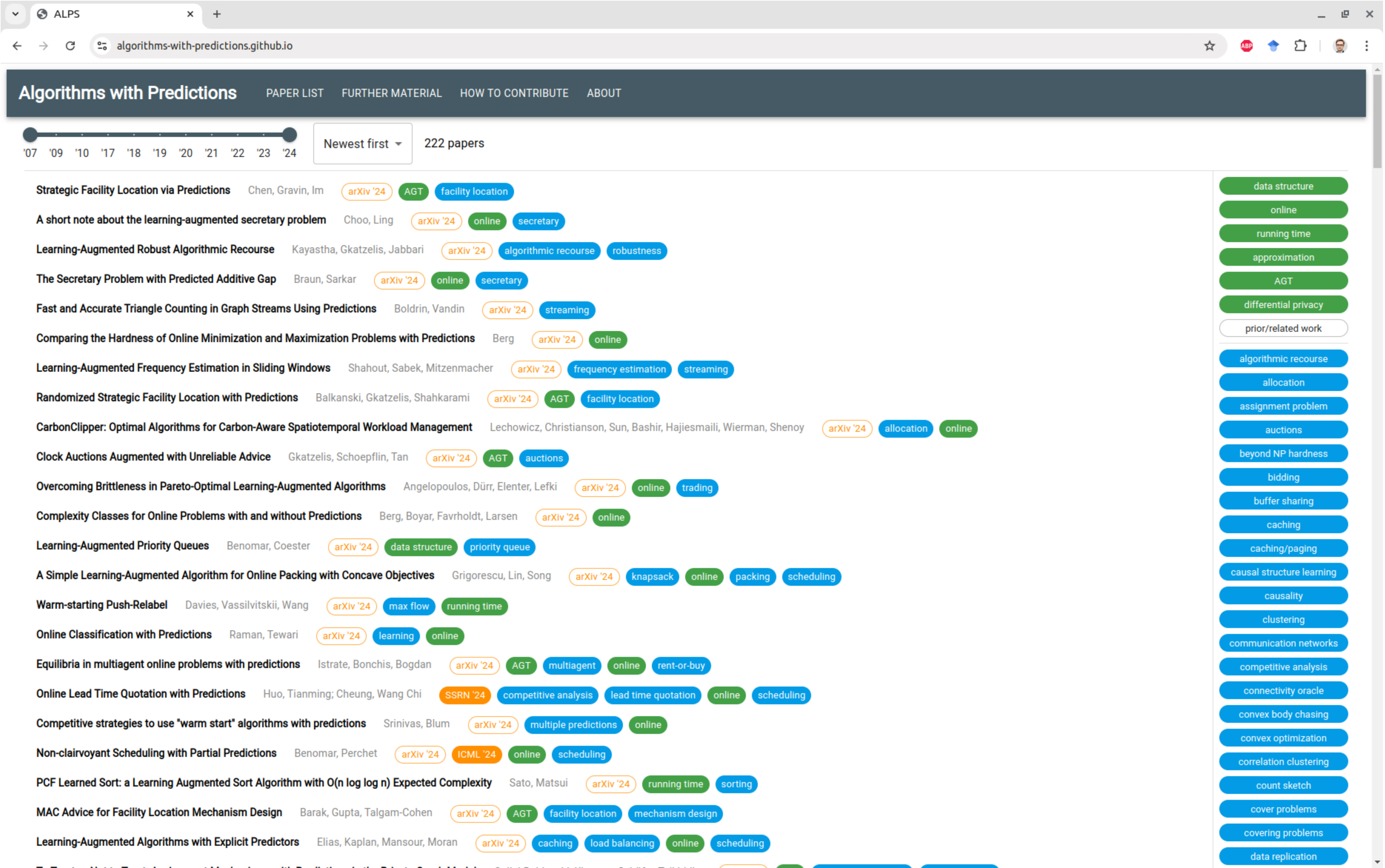
$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} y_v \\ \text{subject to} \quad & y_u + y_v \leqslant c_{u,v} \quad \forall\, (u, v) \in E \end{aligned}$$

Prediction: *dual LP solution*

Result: $\boldsymbol{O(m\sqrt{n} \cdot \min(\sqrt{n}, \eta))}$ time algorithm

(without predictions $O(mn)$ time Hungarian algorithm often used in practice)

algorithms-with-predictions.github.io



Alexander Lindermayr

Approximation algorithms

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

approximation ratio (approximation factor)

ALPS

algorithms-with-predictions.github.io

Algorithms with Predictions

PAPER LIST FURTHER MATERIAL HOW TO CONTRIBUTE ABOUT

'07 '09 '10 '17 '18 '19 '20 '21 '22 '23 '24

Newest first

222 papers

Strategic Facility Location via Predictions	Chen, Gravin, Im	arXiv '24, AGT, facility location
A short note about the learning-augmented secretary problem	Choo, Ling	arXiv '24, online, secretary
Learning-Augmented Robust Algorithmic Recourse	Kayastha, Gkatzelis, Jabbari	arXiv '24, algorithmic recourse, robustness
The Secretary Problem with Predicted Additive Gap	Braun, Sarkar	arXiv '24, online, secretary
Fast and Accurate Triangle Counting in Graph Streams Using Predictions	Boldrin, Vandin	arXiv '24, streaming
Comparing the Hardness of Online Minimization and Maximization Problems with Predictions	Berg	arXiv '24, online
Learning-Augmented Frequency Estimation in Sliding Windows	Shahout, Sabek, Mitzenmacher	arXiv '24, frequency estimation, streaming
Randomized Strategic Facility Location with Predictions	Balkanski, Gkatzelis, Shahkarami	arXiv '24, AGT, facility location
CarbonClipper: Optimal Algorithms for Carbon-Aware Spatiotemporal Workload Management	Lechowicz, Christianson, Sun, Bashir, Hajiesmaili, Wierman, Shenoy	arXiv '24, allocation, online
Clock Auctions Augmented with Unreliable Advice	Gkatzelis, Schoepflin, Tan	arXiv '24, AGT, auctions
Overcoming Brittleness in Pareto-Optimal Learning-Augmented Algorithms	Angelopoulos, Dürr, Elenter, Lefki	arXiv '24, online, trading
Complexity Classes for Online Problems with and without Predictions	Berg, Boyar, Favrholdt, Larsen	arXiv '24, online
Learning-Augmented Priority Queues	Benomar, Coester	arXiv '24, data structure, priority queue
A Simple Learning-Augmented Algorithm for Online Packing with Concave Objectives	Grigorescu, Lin, Song	arXiv '24, knapsack, online, packing, scheduling
Warm-starting Push-Relabel	Davies, Vassilvitskii, Wang	arXiv '24, max flow, running time
Online Classification with Predictions	Raman, Tewari	arXiv '24, learning, online
Equilibria in multiagent online problems with predictions	Istrate, Bonchis, Bogdan	arXiv '24, AGT, multiagent, online, rent-or-buy
Online Lead Time Quotation with Predictions	Huo, Tianming; Cheung, Wang Chi	SSRN '24, competitive analysis, lead time quotation, online, scheduling
Competitive strategies to use "warm start" algorithms with predictions	Srinivas, Blum	arXiv '24, multiple predictions, online
Non-clairvoyant Scheduling with Partial Predictions	Benomar, Perchet	arXiv '24, ICML '24, online, scheduling
PCF Learned Sort: a Learning Augmented Sort Algorithm with O(n log log n) Expected Complexity	Sato, Matsui	arXiv '24, running time, sorting
MAC Advice for Facility Location Mechanism Design	Barak, Gupta, Talgam-Cohen	arXiv '24, AGT, facility location, mechanism design
Learning-Augmented Algorithms with Explicit Predictors	Elias, Kaplan, Mansour, Moran	arXiv '24, caching, load balancing, online, scheduling

data structure
online
running time
approximation
AGT
differential privacy
prior/related work

algorithmic recourse
allocation
assignment problem
auctions
beyond NP hardness
bidding
buffer sharing
caching
caching/paging
causal structure learning
causality
clustering
communication networks
competitive analysis
connectivity oracle
convex body chasing
convex optimization
correlation clustering
count sketch
cover problems
covering problems
data replication

ALPS

algorithms-with-predictions.github.io

Algorithms with Predictions

PAPER LIST FURTHER MATERIAL HOW TO CONTRIBUTE ABOUT

'07 '09 '10 '17 '18 '19 '20 '21 '22 '23 '24

Newest first

6 papers

RESET

Max-Cut with ε-Accurate Predictions	Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi	arXiv '24, approximation, max-cut
Parsimonious Learning-Augmented Approximations for Dense Instances of NP-hard Problems	Bampis, Escoffier, Xefteris	arXiv '24, ICML '24, approximation, graph problems, running time
Learning-augmented Maximum Independent Set	Braverman, Dharangutte, Shah, Wang	APPROX/RANDOM '24, arXiv '24, approximation, graph problems
Improved Learning-augmented Algorithms for k-means and k-medians Clustering	Thy Nguyen, Anamay Chaturvedi, Huy Nguyen	arXiv '22, ICLR '23, approximation, clustering
Approximate Cluster Recovery from Noisy Labels	Gamlath, Lattanzi, Norouzi-Fard, Svensson	COLT '22, approximation, clustering
Learning-Augmented k-means Clustering	Ergun, Feng, Silwal, Woodruff, Zhou	arXiv '21, ICLR '22, approximation, beyond NP hardness, clustering, running time

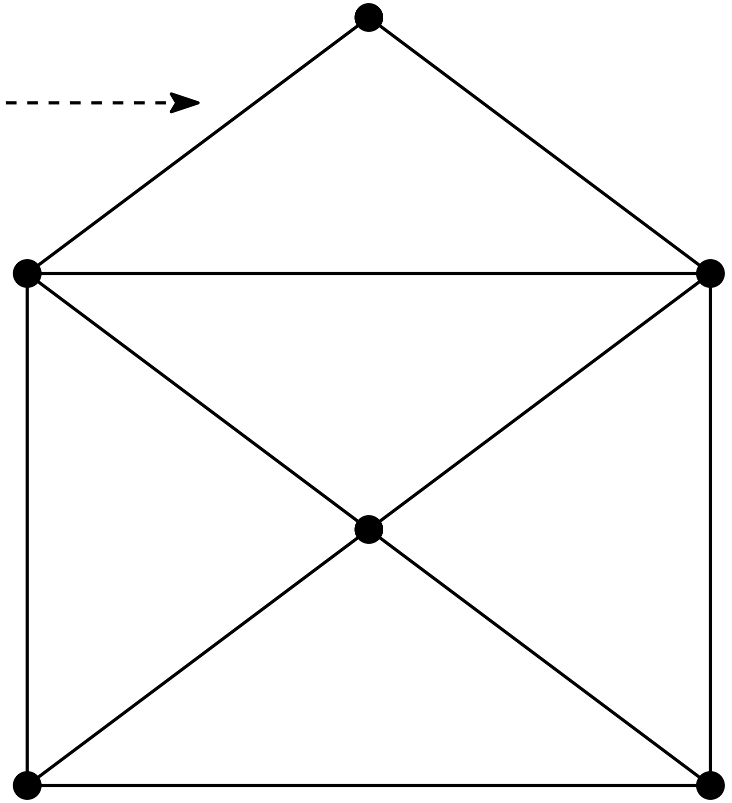
data structure
online
running time
approximation
AGT
differential privacy
prior/related work

algorithmic recourse
allocation
assignment problem
auctions
beyond NP hardness
bidding
buffer sharing
caching
caching/paging
causal structure learning
causality
clustering
communication networks
competitive analysis
connectivity oracle
convex body chasing
convex optimization
correlation clustering
count sketch
cover problems
covering problems
data replication

Example: the Steiner Tree problem

Input:

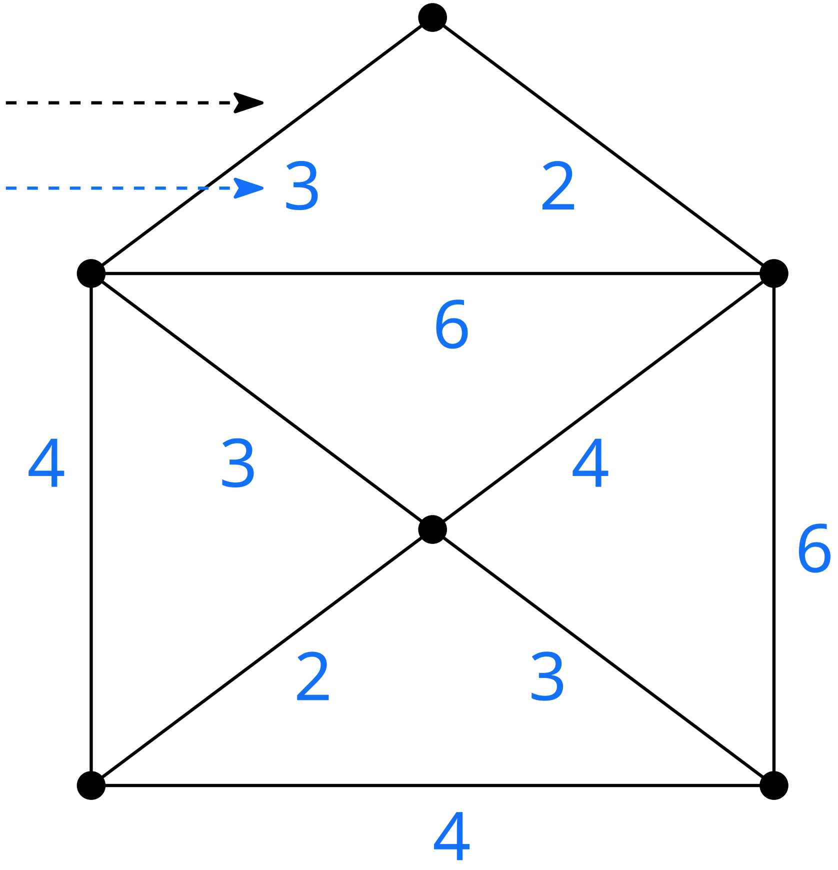
- undirected **graph** $G = (V, E)$



Example: the Steiner Tree problem

Input:

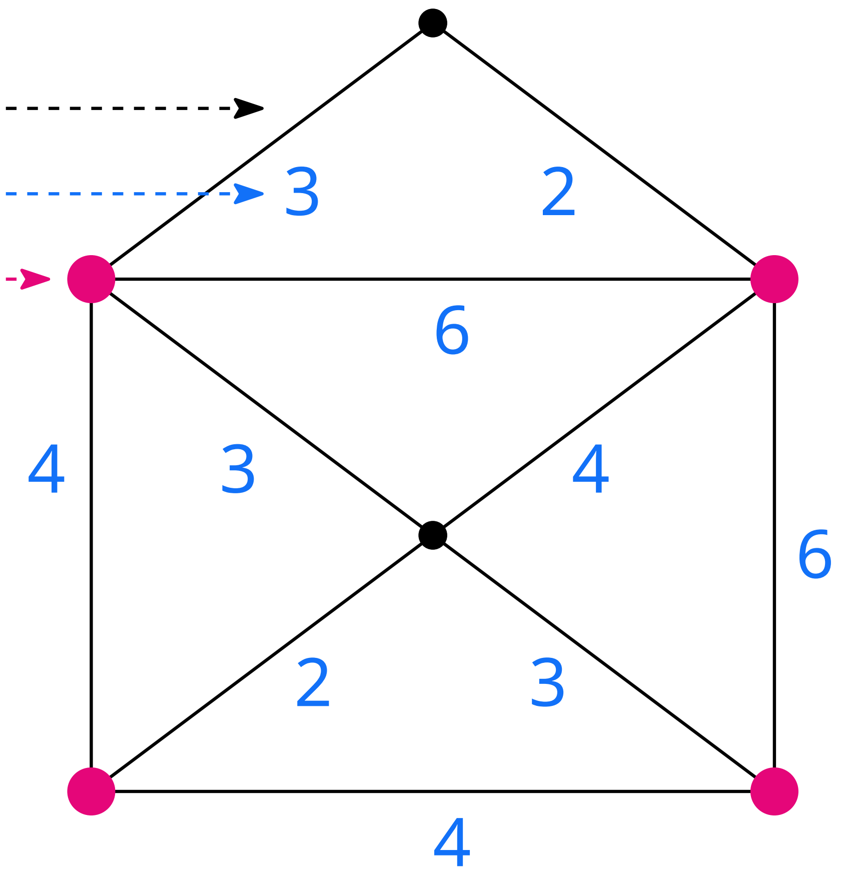
- undirected **graph** $G = (V, E)$
- edge **weights** $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$



Example: the Steiner Tree problem

Input:

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



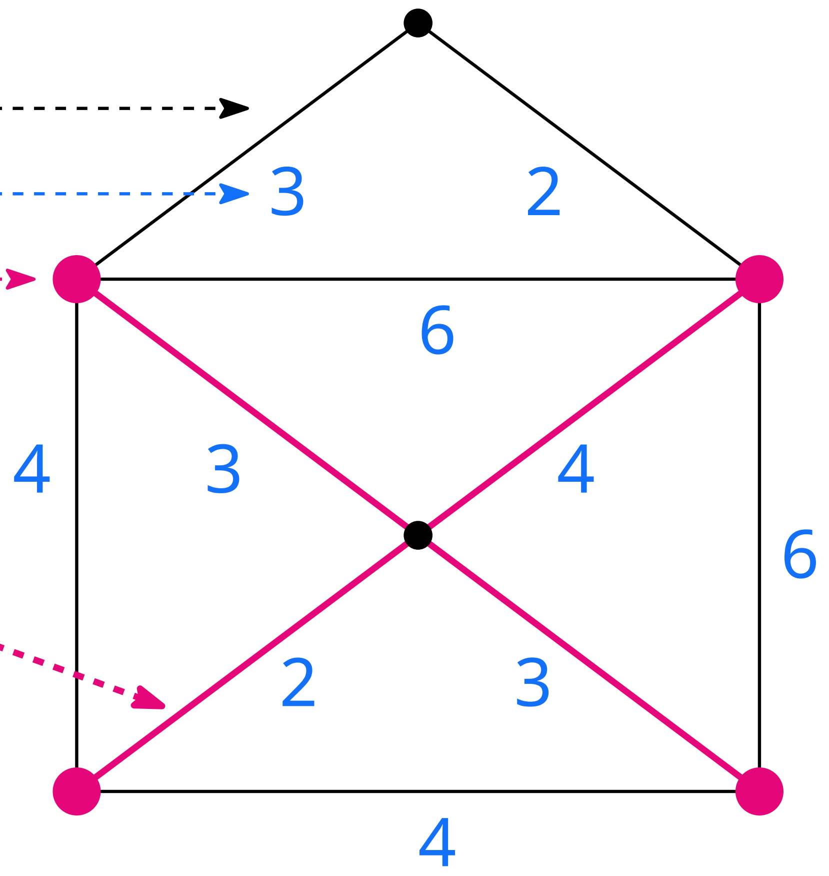
Example: the Steiner Tree problem

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



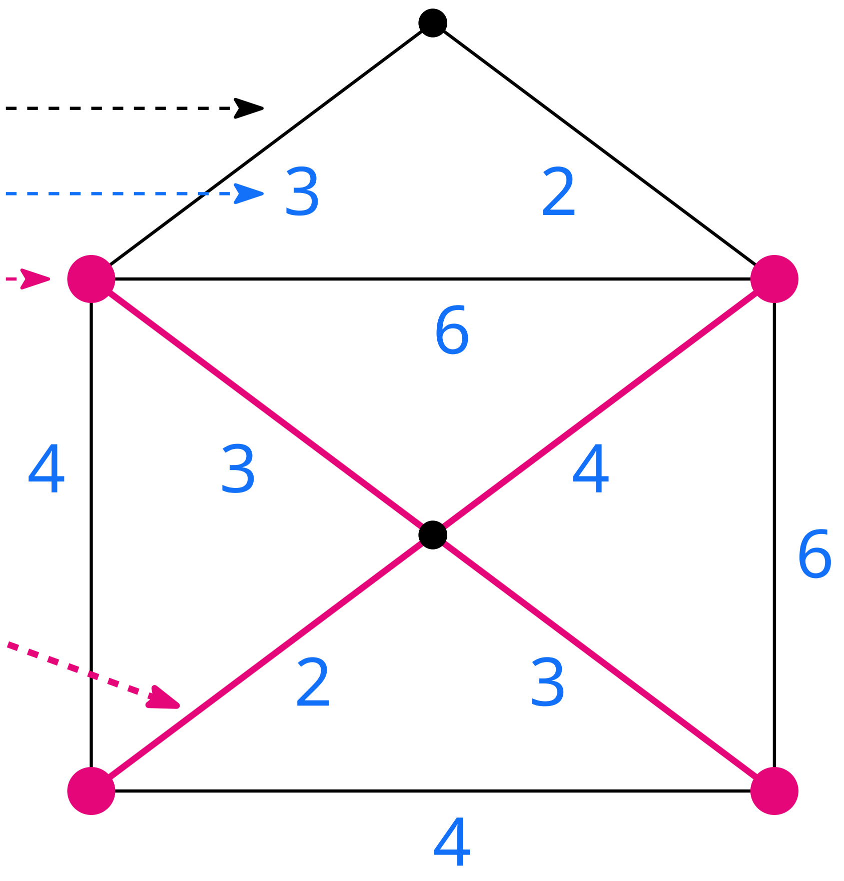
Example: the Steiner Tree problem

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



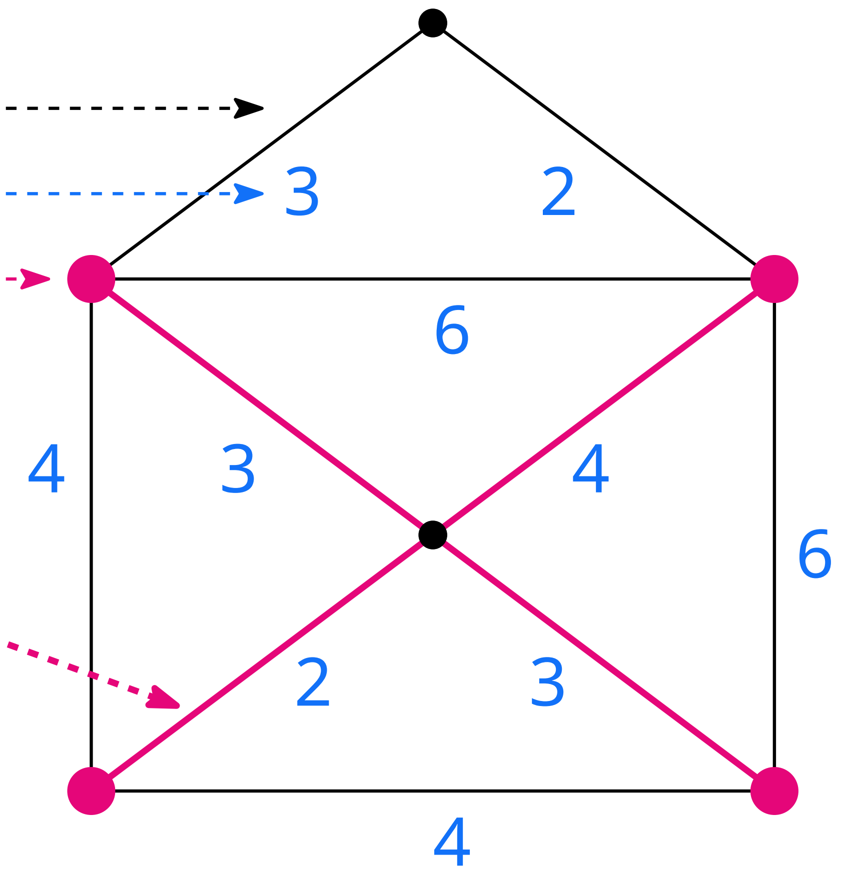
What is known?

- 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]

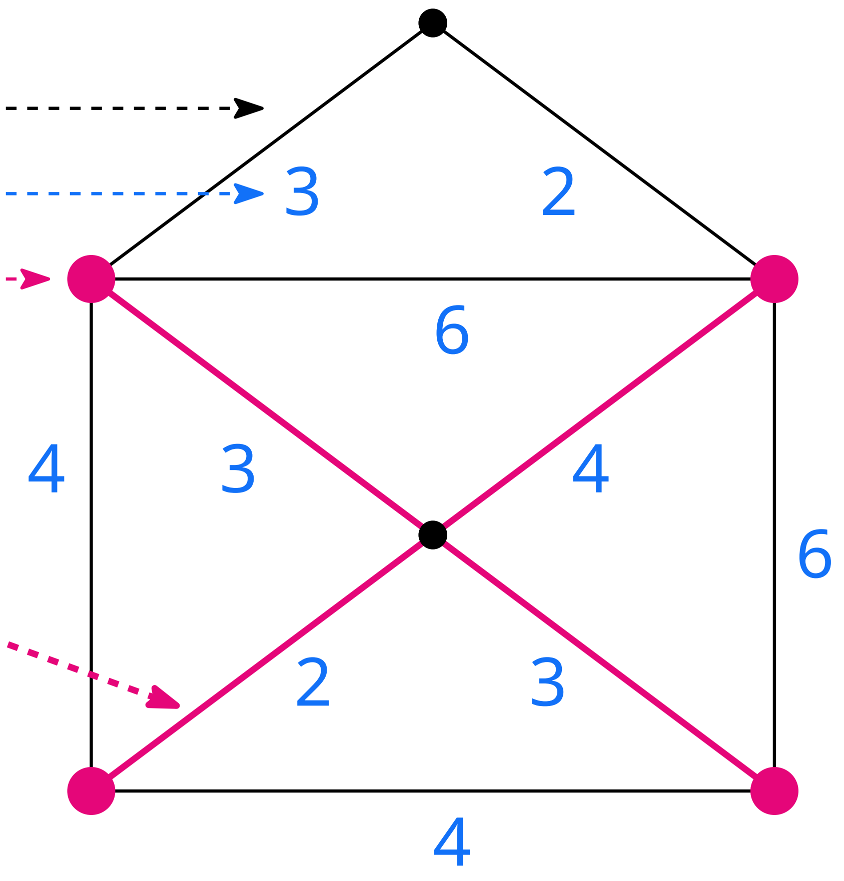
Example: the Steiner Tree problem

Input:

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

Output:

- min weight subgraph of G spanning T



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

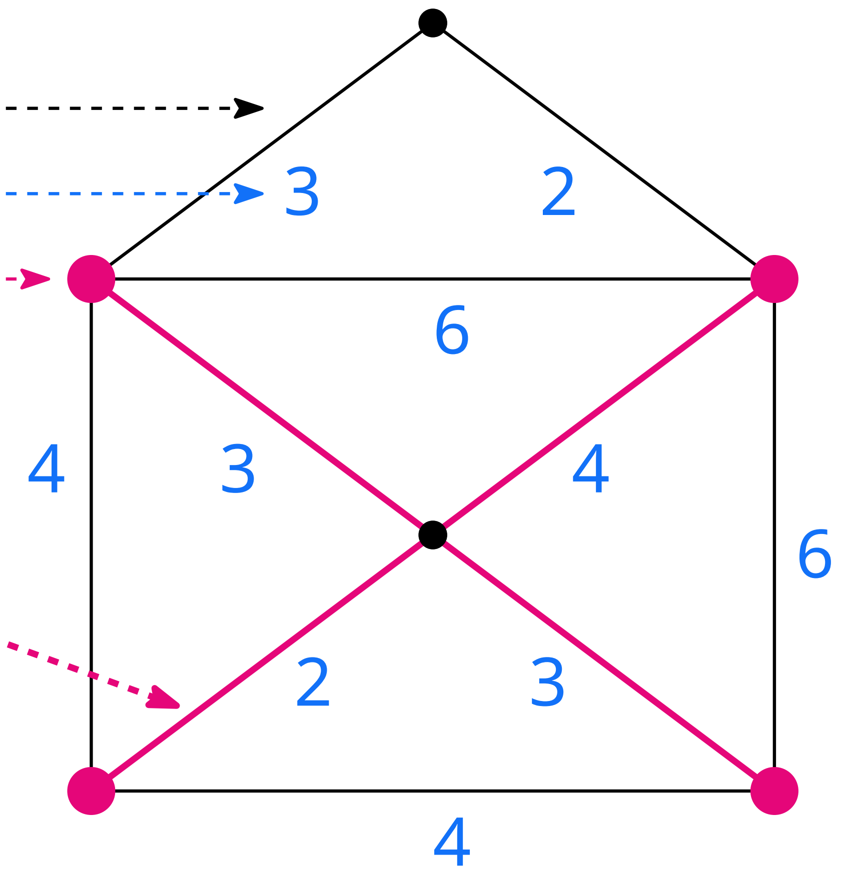
Example: the Steiner Tree problem

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



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

[Antoniadis, Eliáš, **P.**, Venzin ’24]

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**

[Antoniadis, Eliáš, **P.**, Venzin '24]

Steiner Tree with predictions

Input:

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

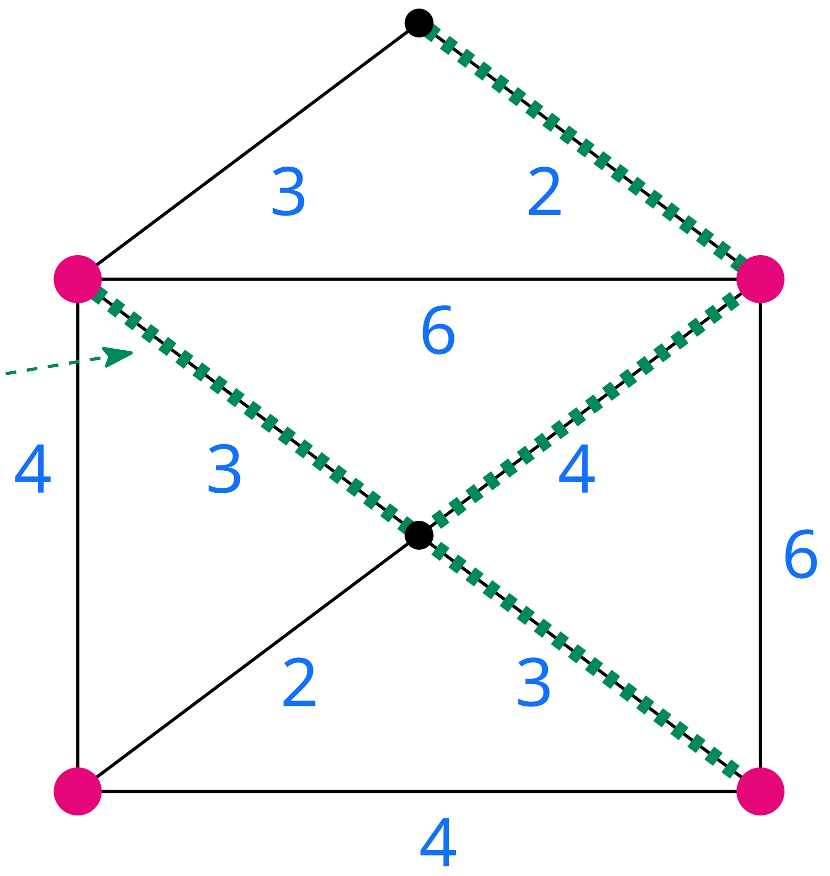
Prediction:

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

(not necessarily feasible)

Output:

- min weight subgraph of G spanning T



Steiner Tree with predictions

Input:

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

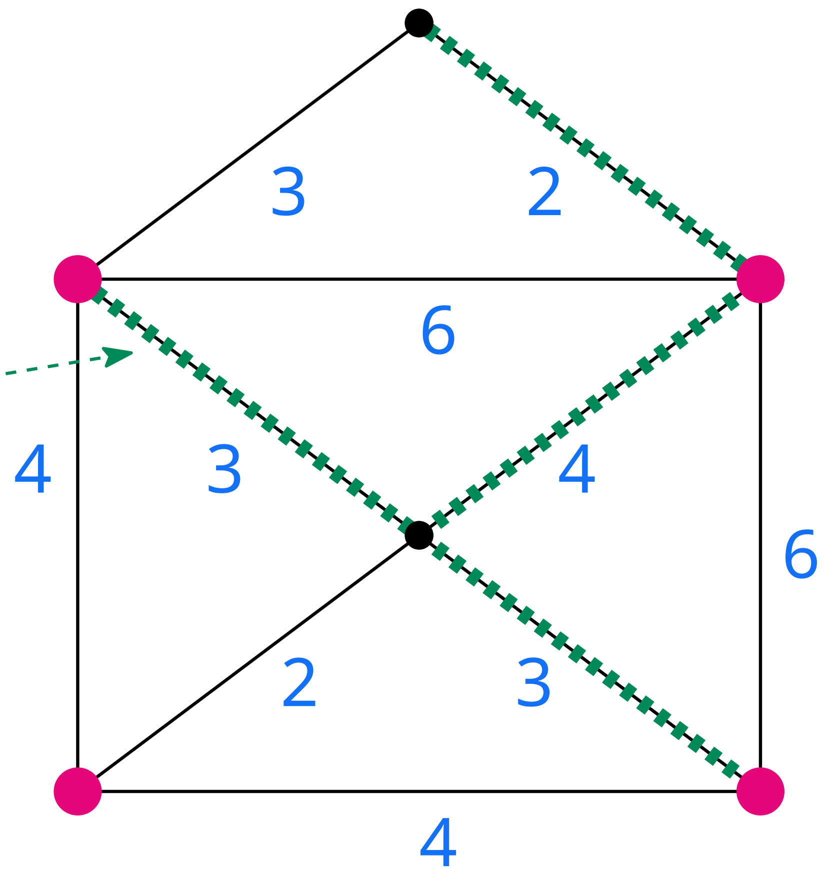
Prediction:

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

(not necessarily feasible)

Output:

- min weight subgraph of G spanning T



Our result:

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

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

Steiner Tree with predictions

Input:

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

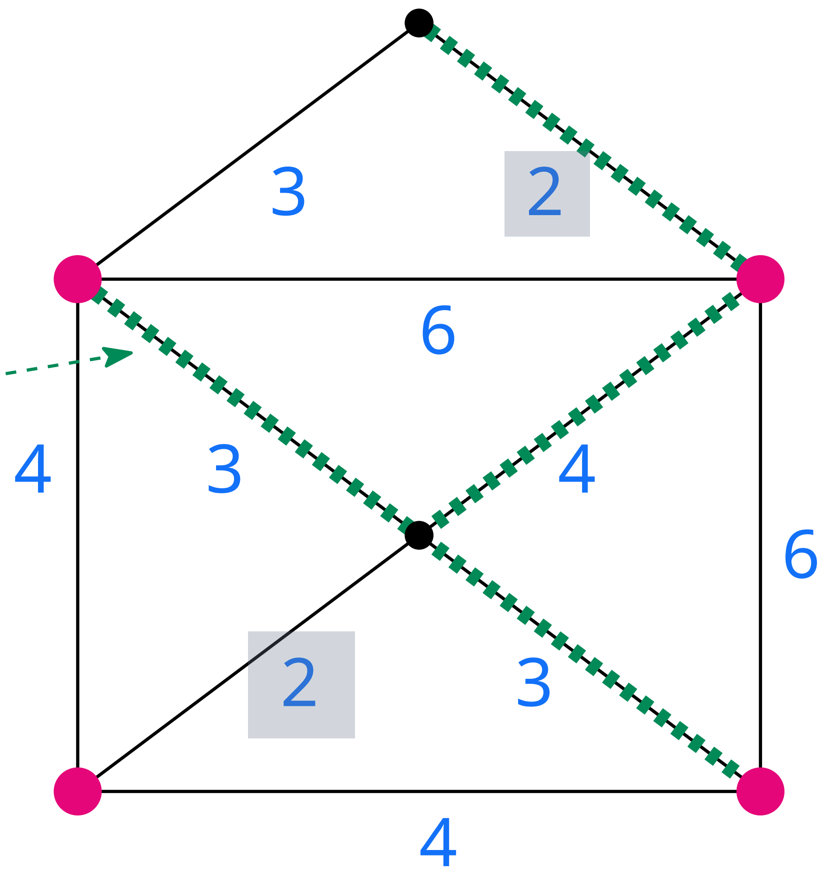
Prediction:

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

(not necessarily feasible)

Output:

- ▶ min weight subgraph of G spanning T



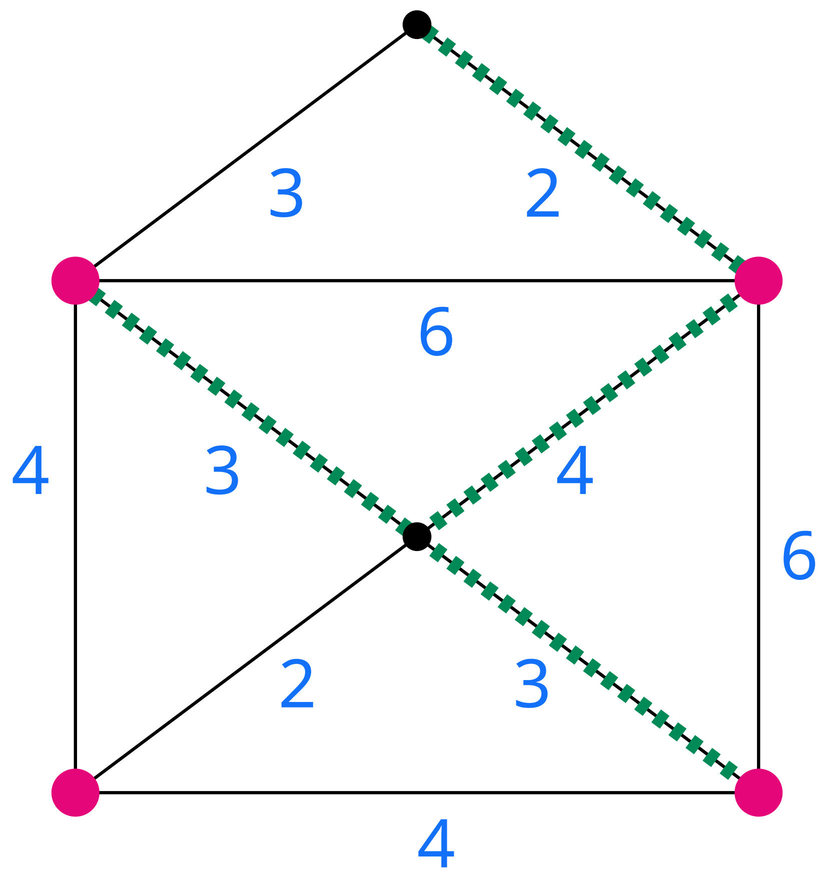
Our result:

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

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

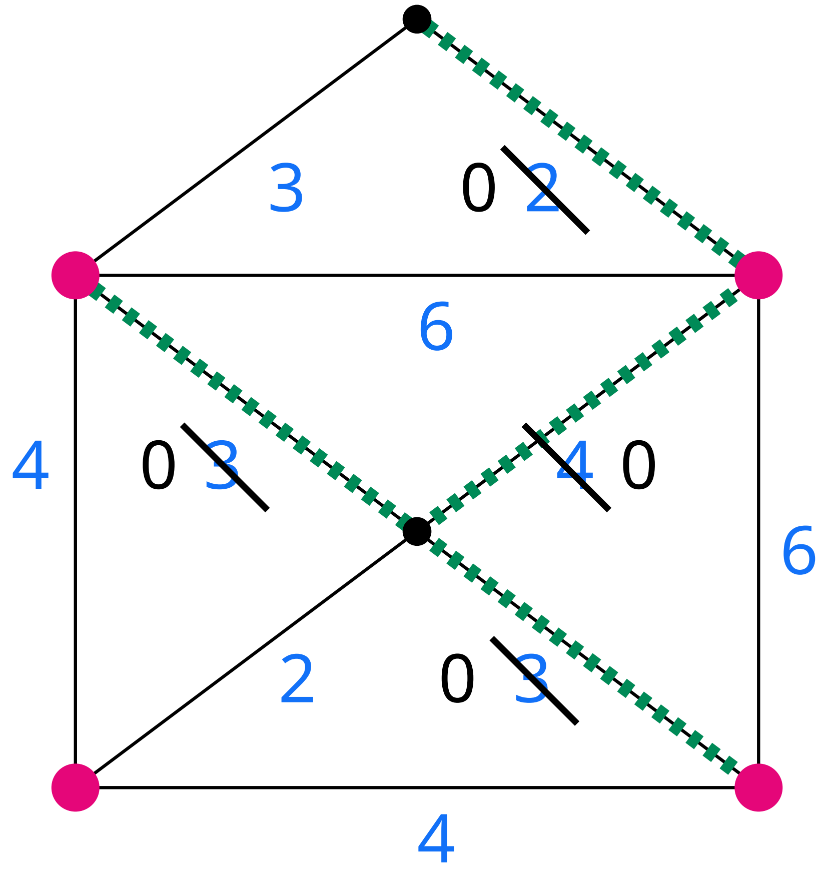
Our learning-augmented algorithm for Steiner Tree

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



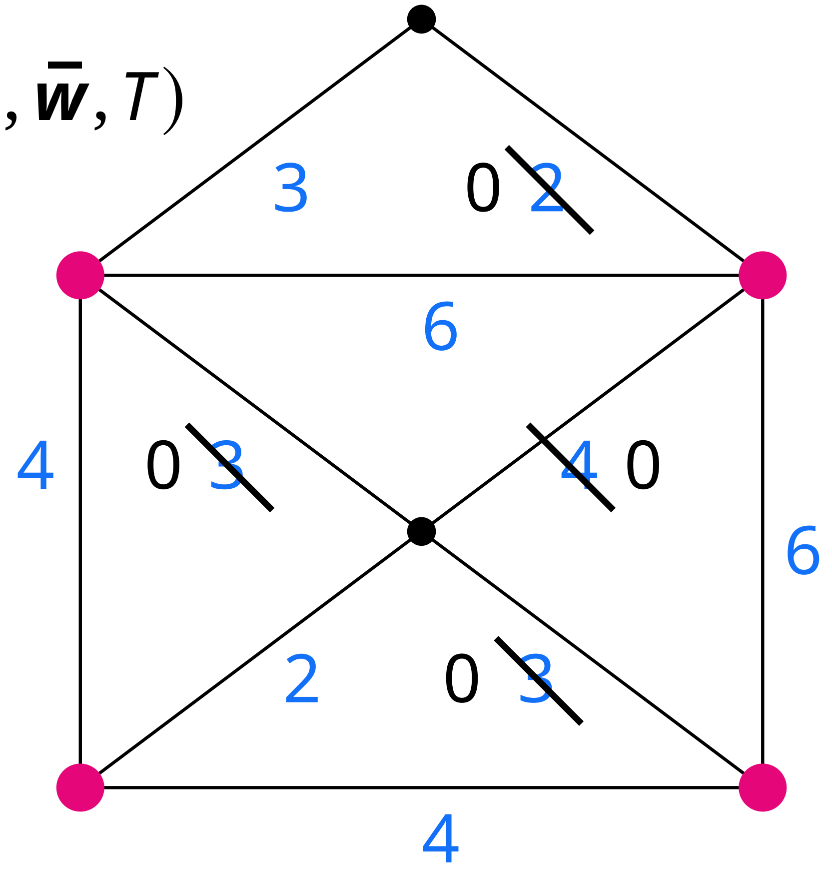
Our learning-augmented algorithm for Steiner Tree

- 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 $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$



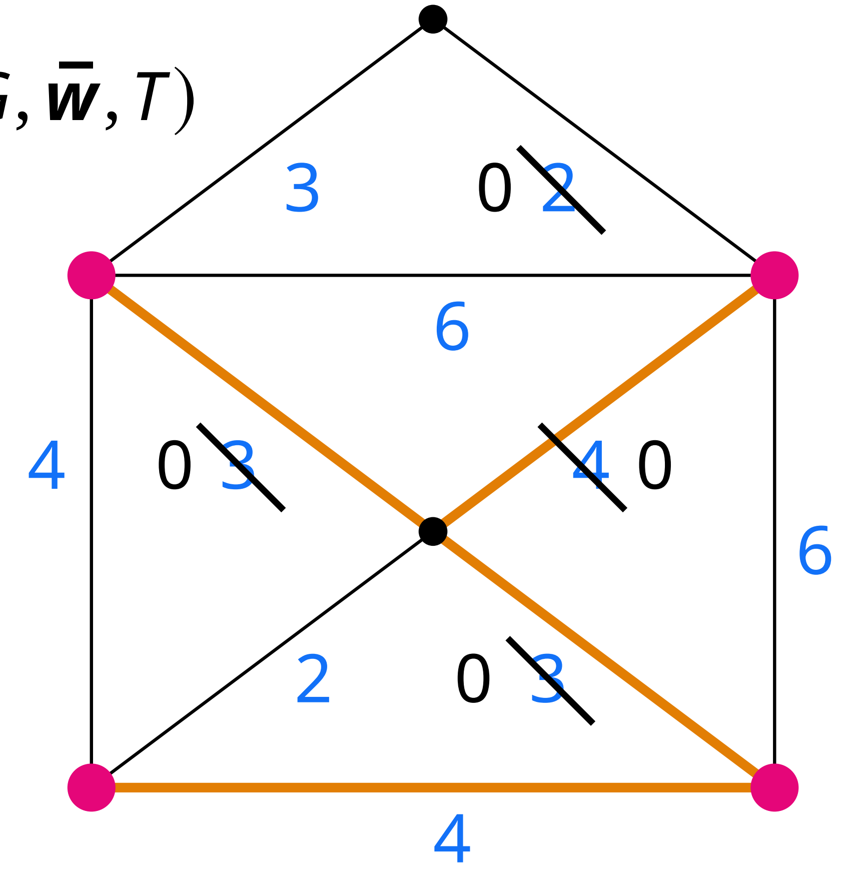
Our learning-augmented algorithm for Steiner Tree

- 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 $(G, \bar{\boldsymbol{w}}, T)$



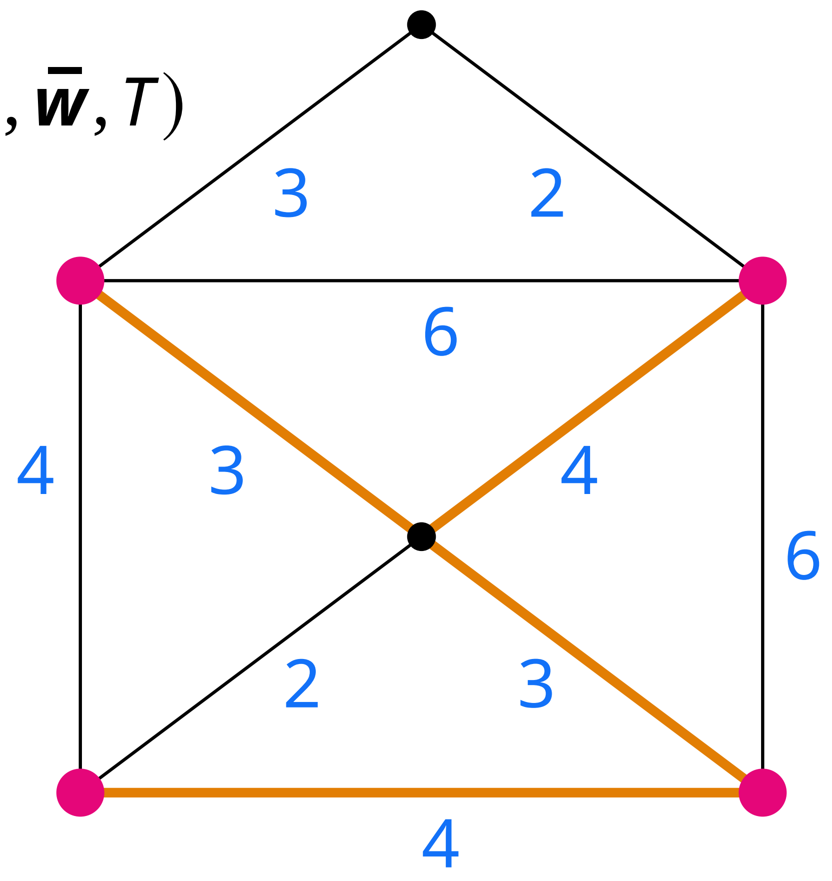
Our learning-augmented algorithm for Steiner Tree

- 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 $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$



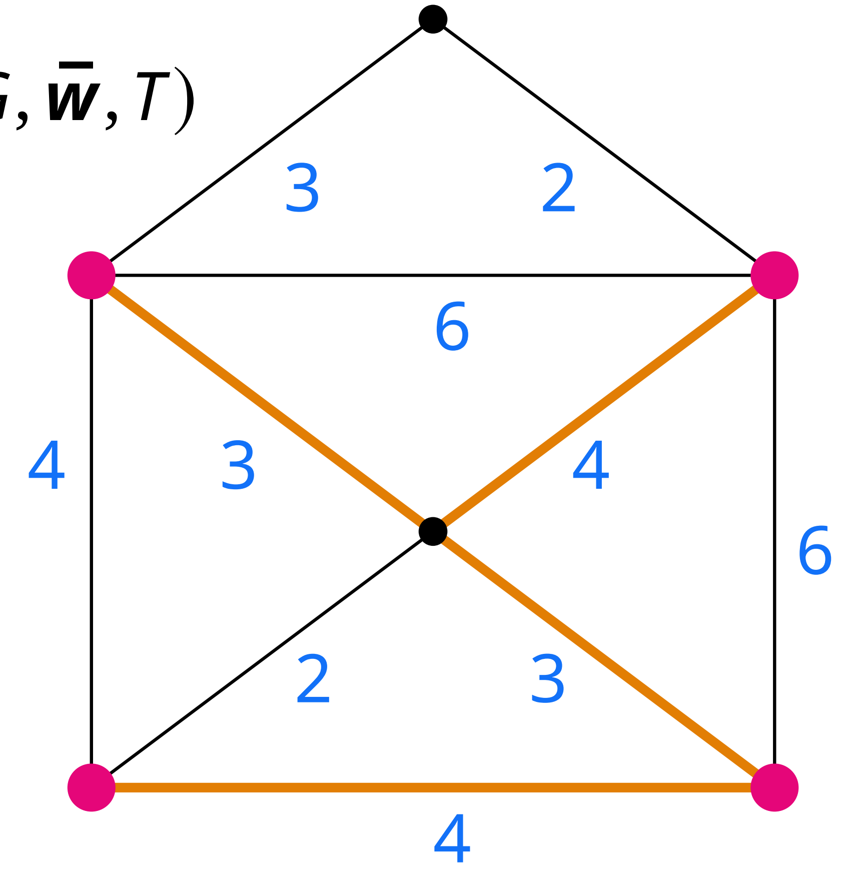
Our learning-augmented algorithm for Steiner Tree

- 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 $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned



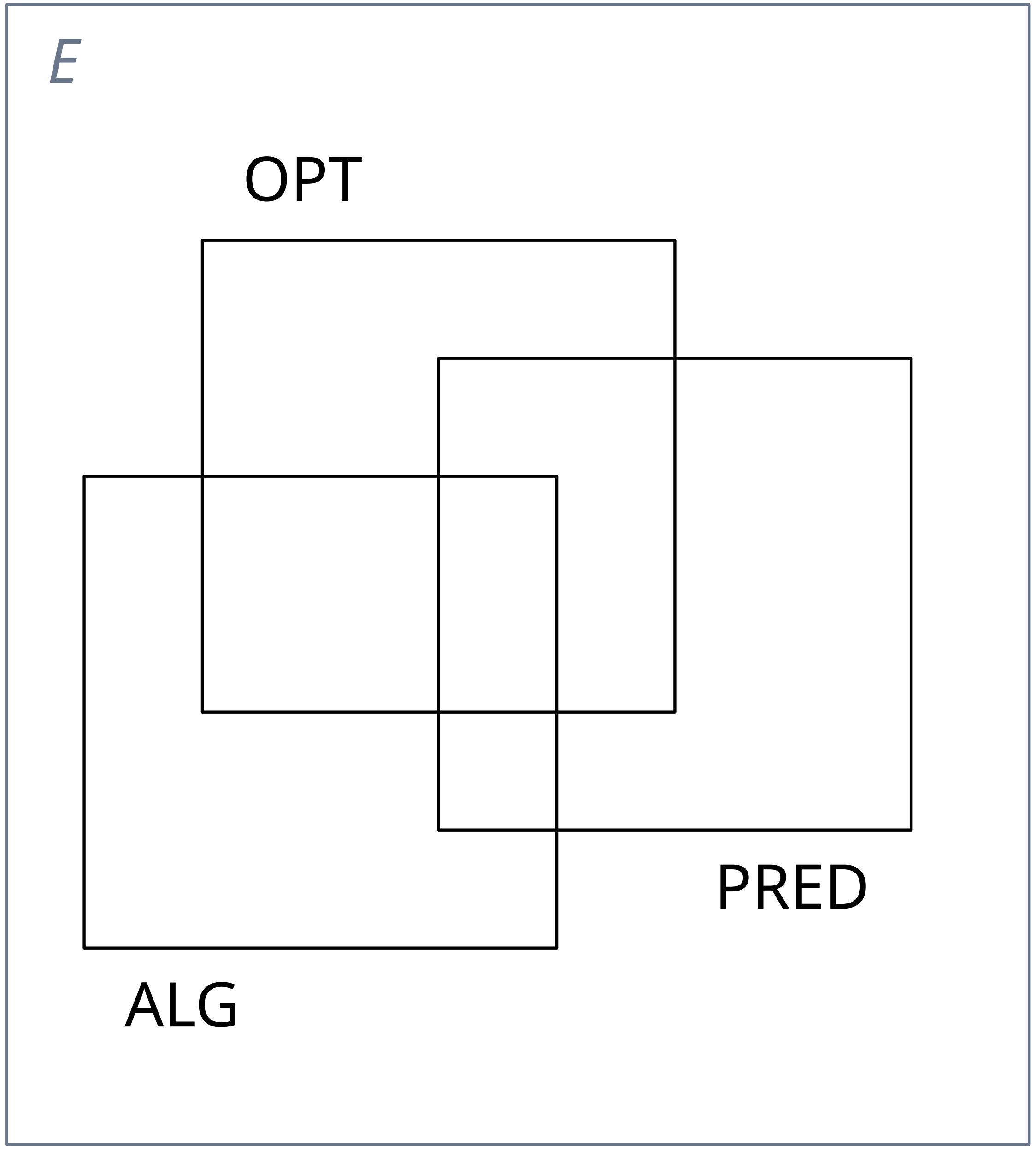
Our learning-augmented algorithm for Steiner Tree

- 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 $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \mathbf{0}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned

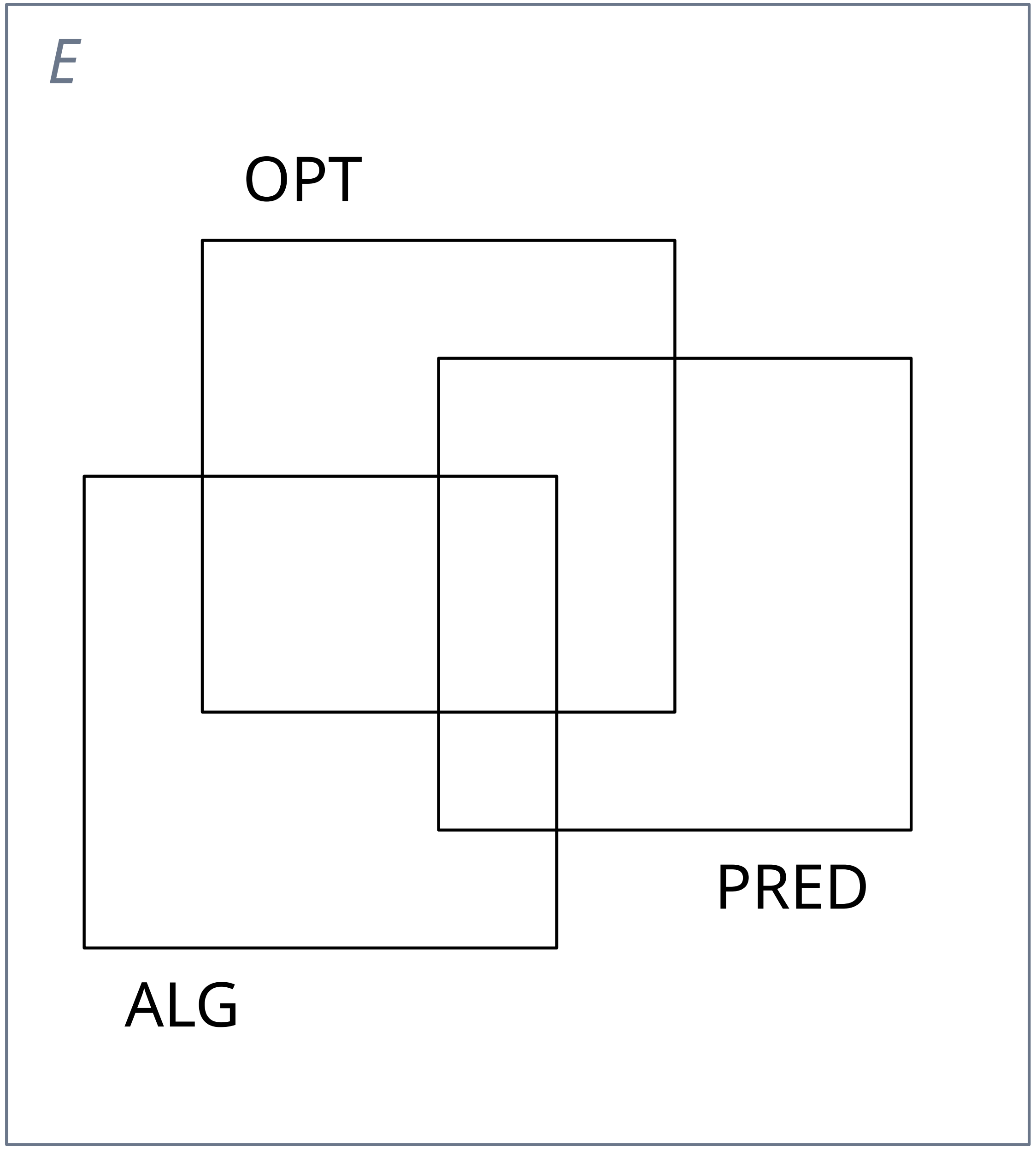


Claim: $w(\text{ALG}) \leqslant (1 + \eta/\text{OPT}) \cdot \text{OPT} = \text{OPT} + \eta$

The proof

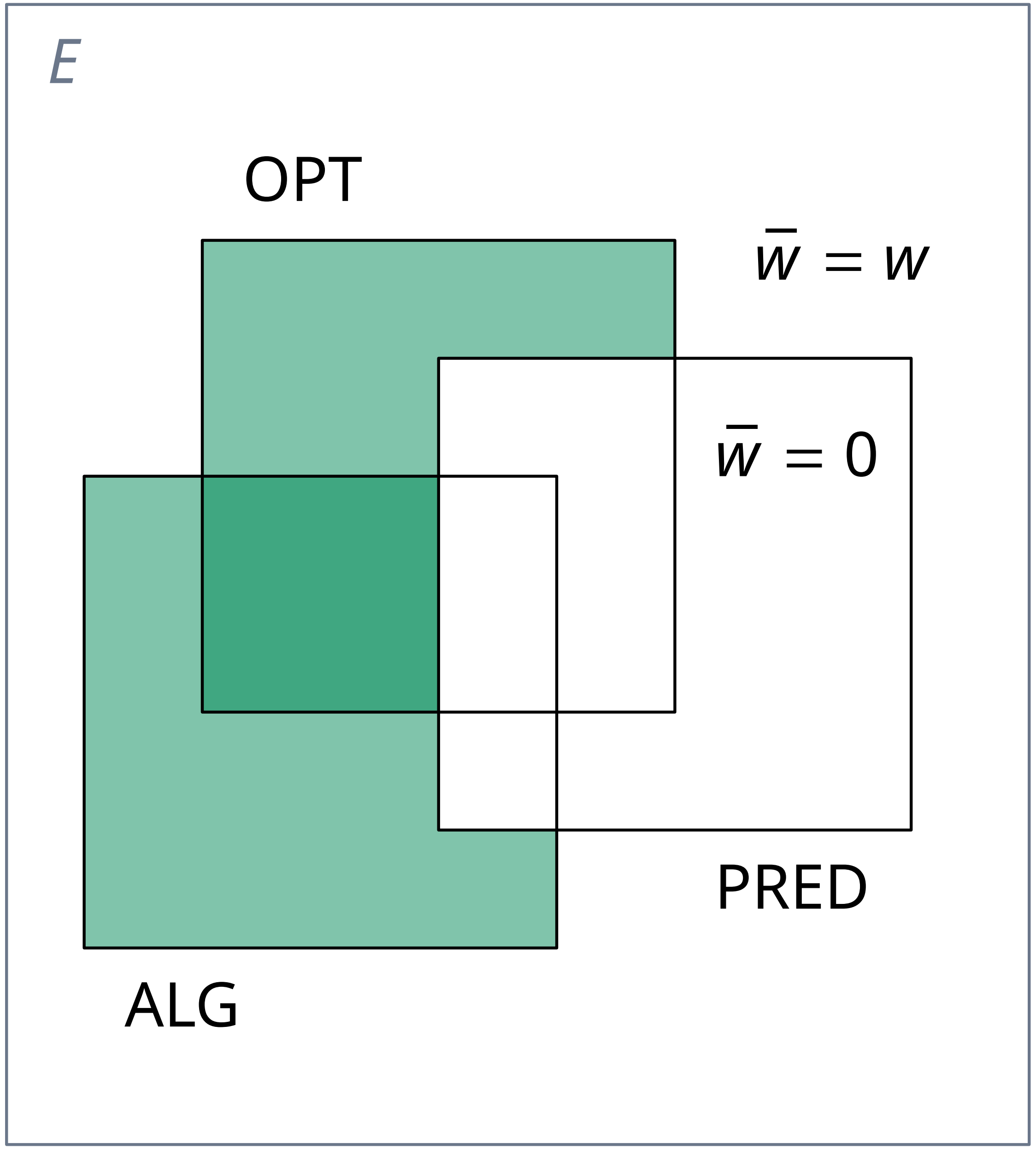


The proof



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

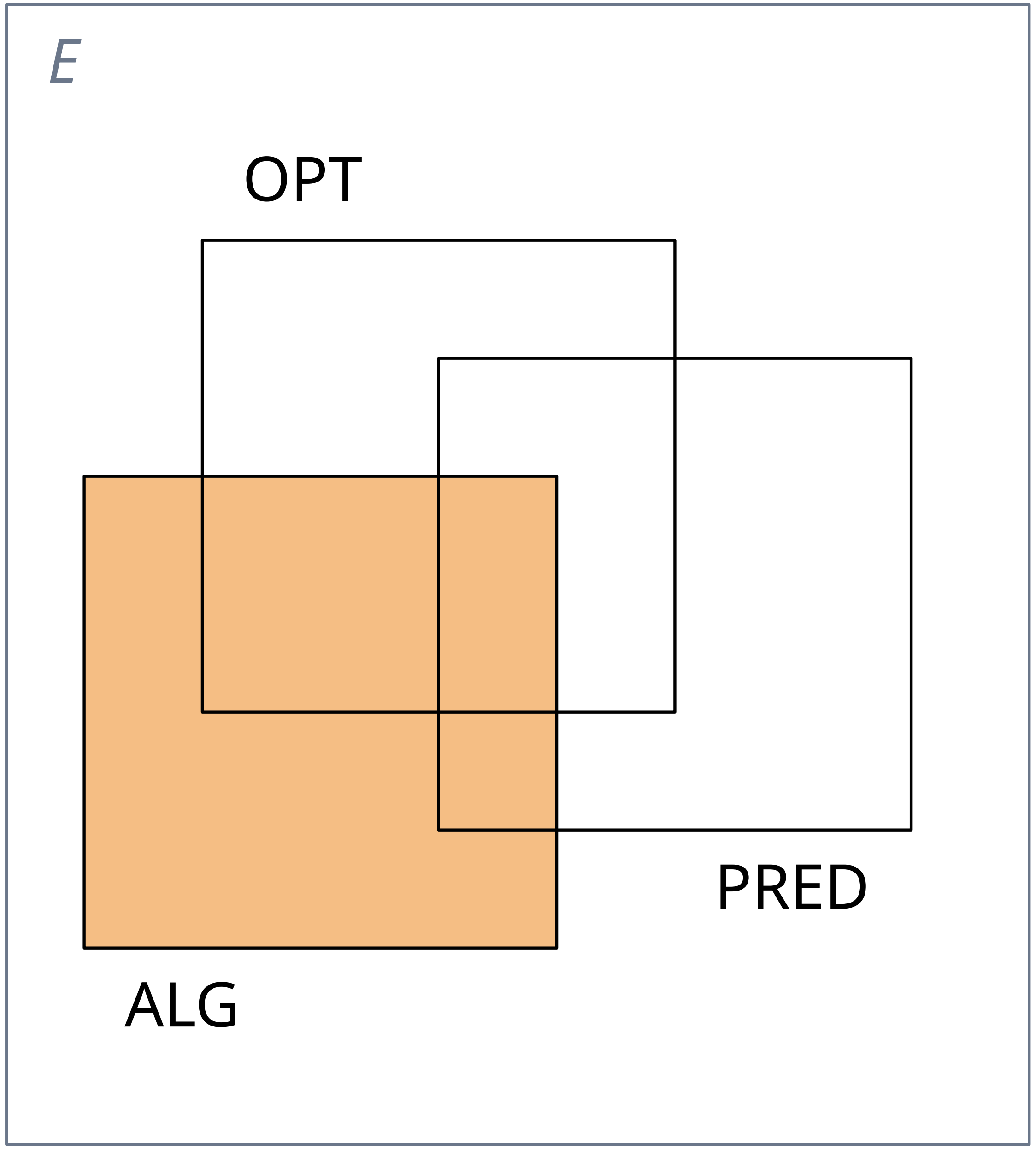
The proof



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

$$w(\mathrm{ALG} \setminus \mathrm{PRED}) \leqslant 2 \cdot w(\mathrm{OPT} \setminus \mathrm{PRED})$$

The proof

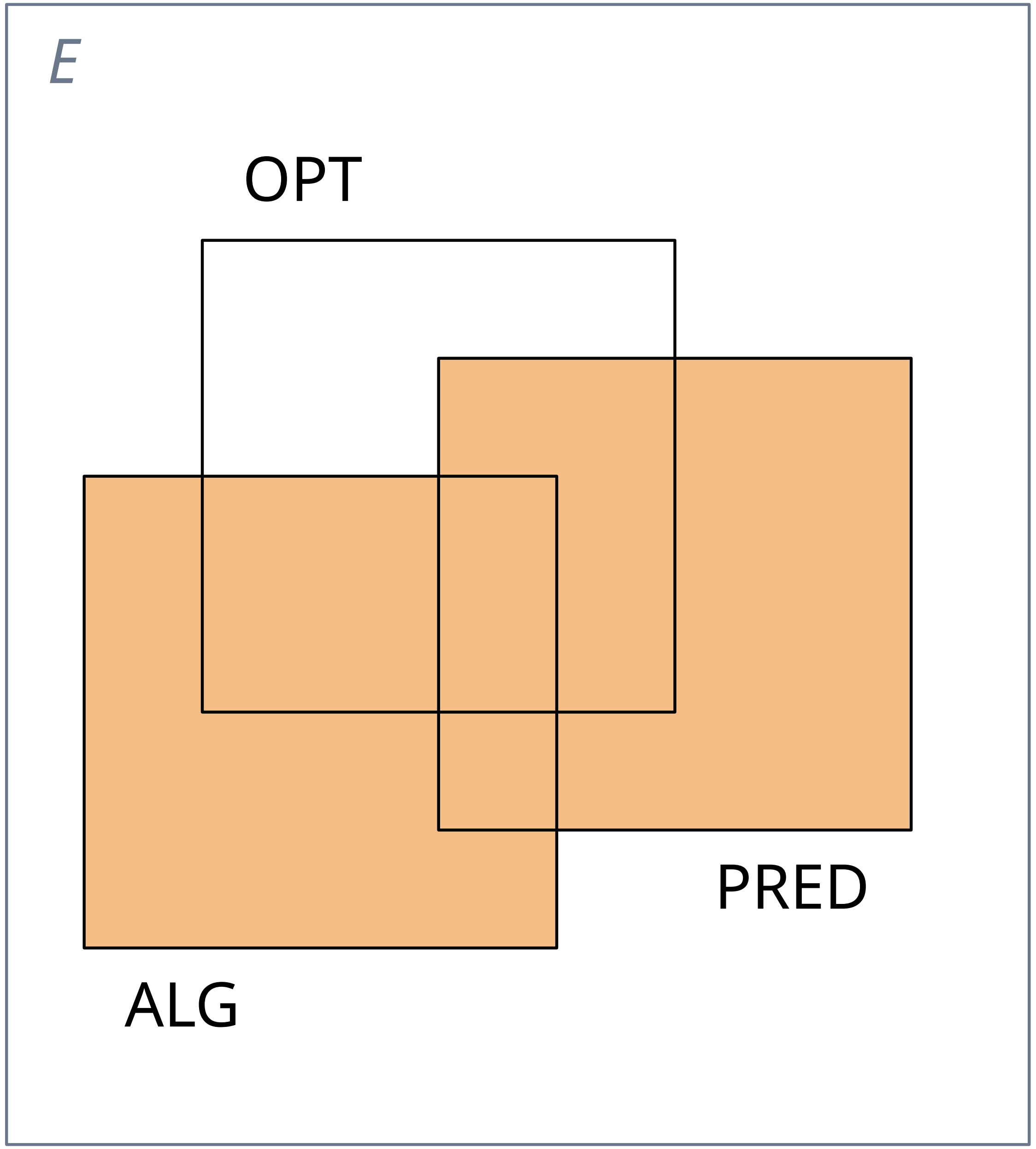


$$\bar{w}(\mathrm{ALG}) \leqslant 2 \cdot \bar{w}(\mathrm{OPT})$$

$$w(\mathrm{ALG} \setminus \mathrm{PRED}) \leqslant 2 \cdot w(\mathrm{OPT} \setminus \mathrm{PRED})$$

$$w(\mathrm{ALG})$$

The proof

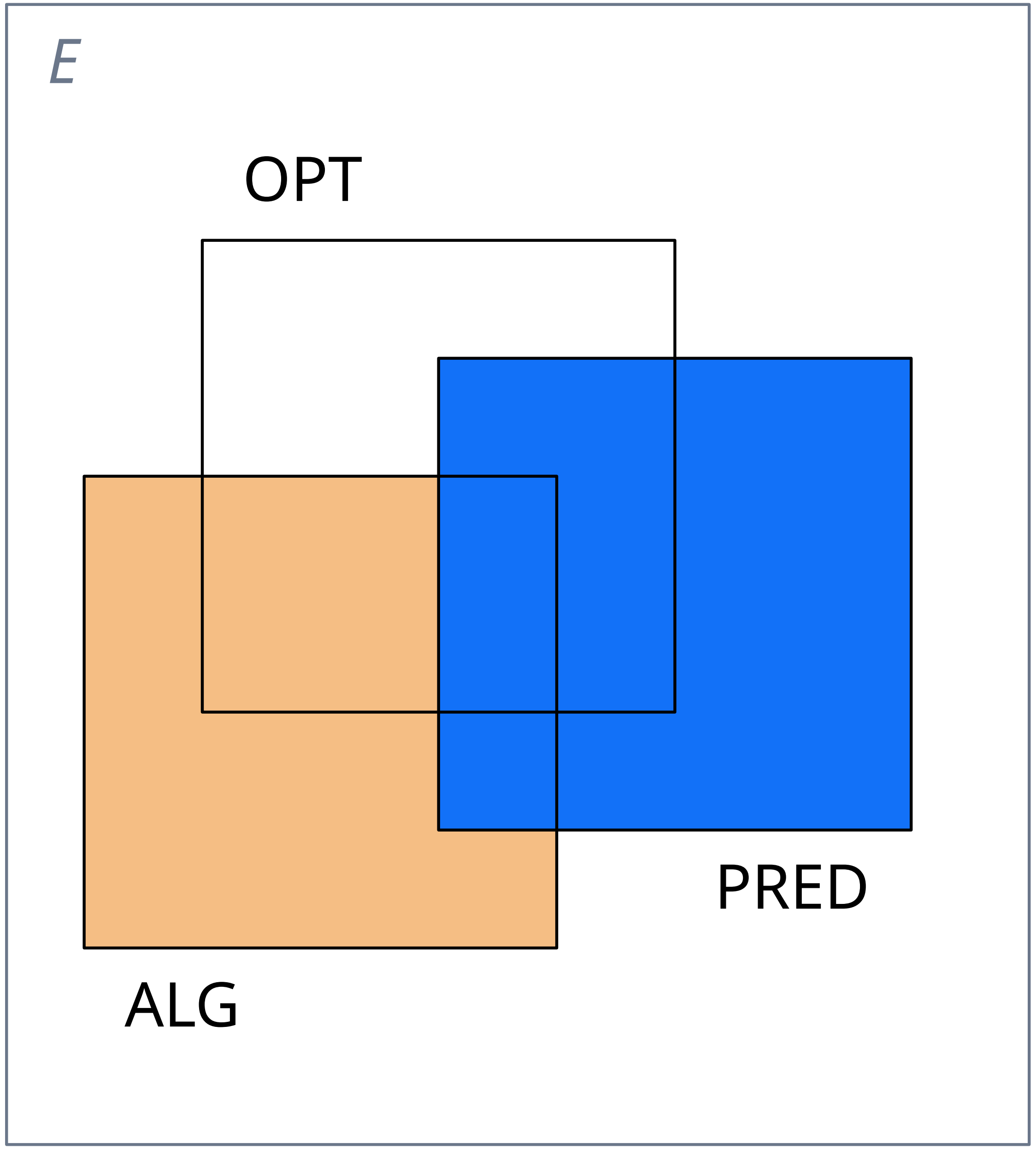


$$\bar{w}(\text{ALG}) \leqslant 2 \cdot \bar{w}(\text{OPT})$$

$$w(\text{ALG} \setminus \text{PRED}) \leqslant 2 \cdot w(\text{OPT} \setminus \text{PRED})$$

$$w(\text{ALG}) \leqslant w(\text{ALG} \cup \text{PRED})$$

The proof



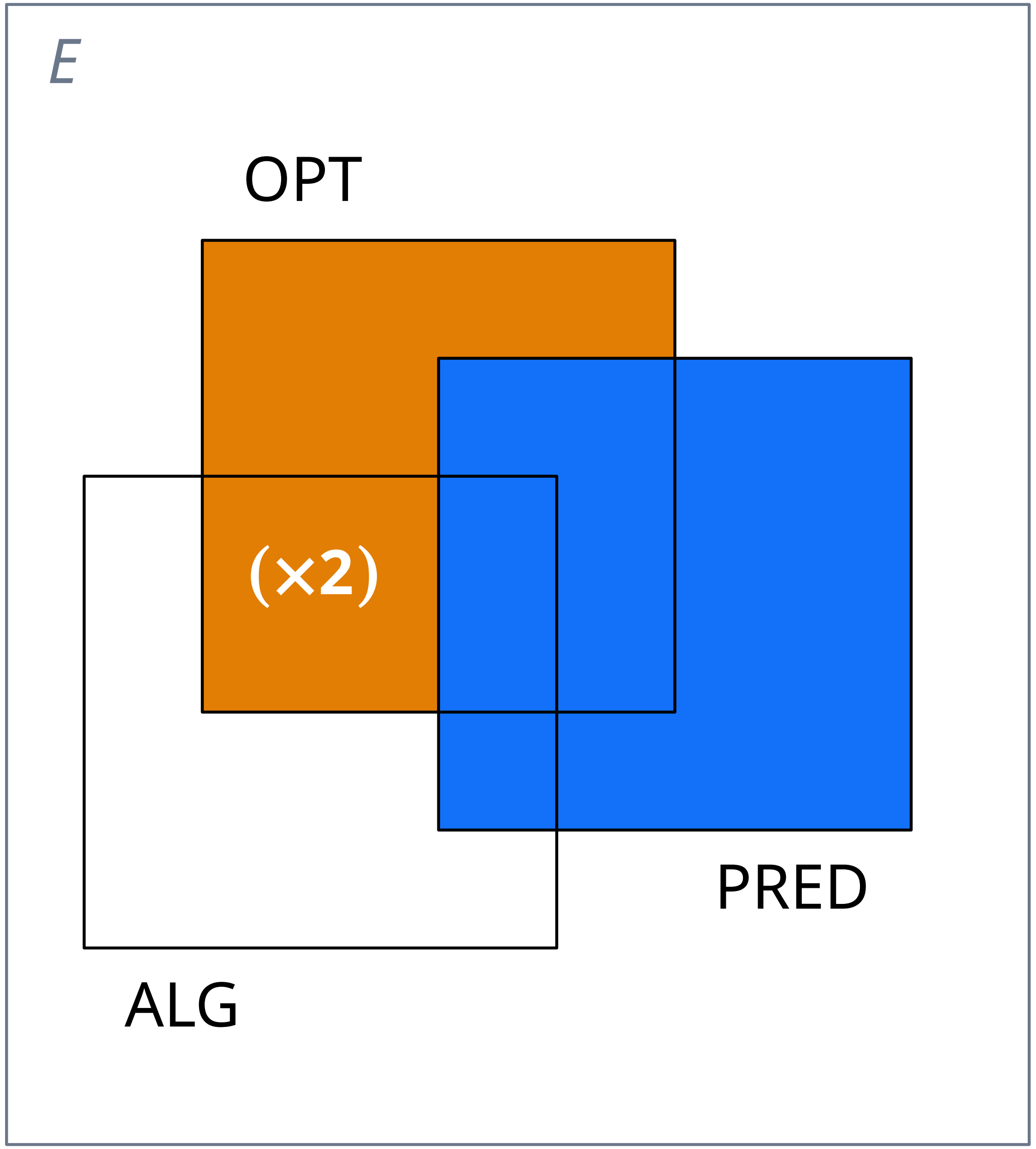
$$\bar{w}(\text{ALG}) \leqslant 2 \cdot \bar{w}(\text{OPT})$$

$$w(\text{ALG} \setminus \text{PRED}) \leqslant 2 \cdot w(\text{OPT} \setminus \text{PRED})$$

$$w(\text{ALG}) \leqslant w(\text{ALG} \cup \text{PRED})$$

$$= w(\text{PRED}) + w(\text{ALG} \setminus \text{PRED})$$

The proof

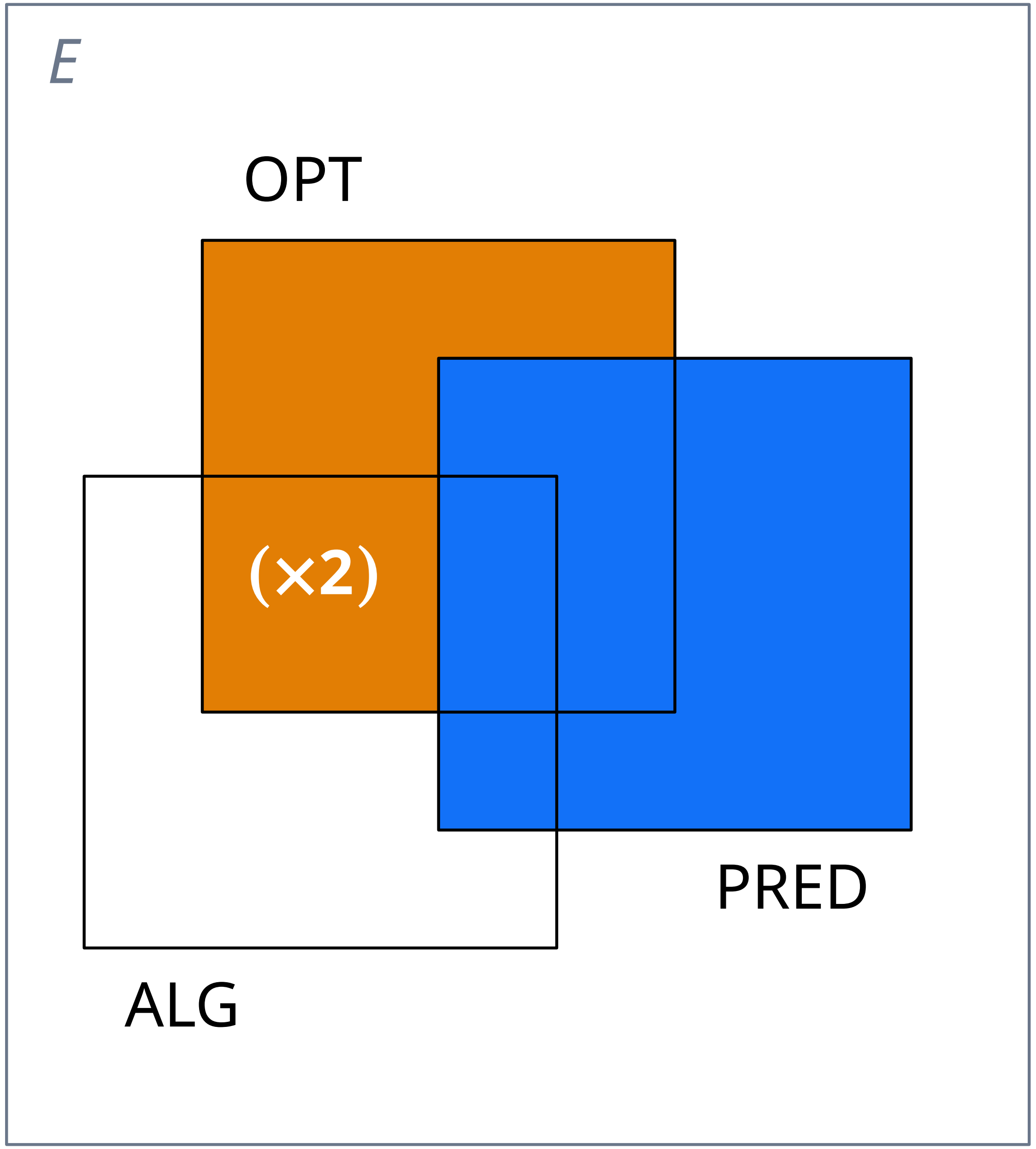


$$\bar{w}(\mathrm{ALG}) \leqslant 2 \cdot \bar{w}(\mathrm{OPT})$$

$$w(\mathrm{ALG} \setminus \mathrm{PRED}) \leqslant 2 \cdot w(\mathrm{OPT} \setminus \mathrm{PRED})$$

$$\begin{aligned} w(\mathrm{ALG}) &\leqslant w(\mathrm{ALG} \cup \mathrm{PRED}) \\ &= w(\mathrm{PRED}) + w(\mathrm{ALG} \setminus \mathrm{PRED}) \\ &\leqslant w(\mathrm{PRED}) + 2 \cdot w(\mathrm{OPT} \setminus \mathrm{PRED}) \end{aligned}$$

The proof

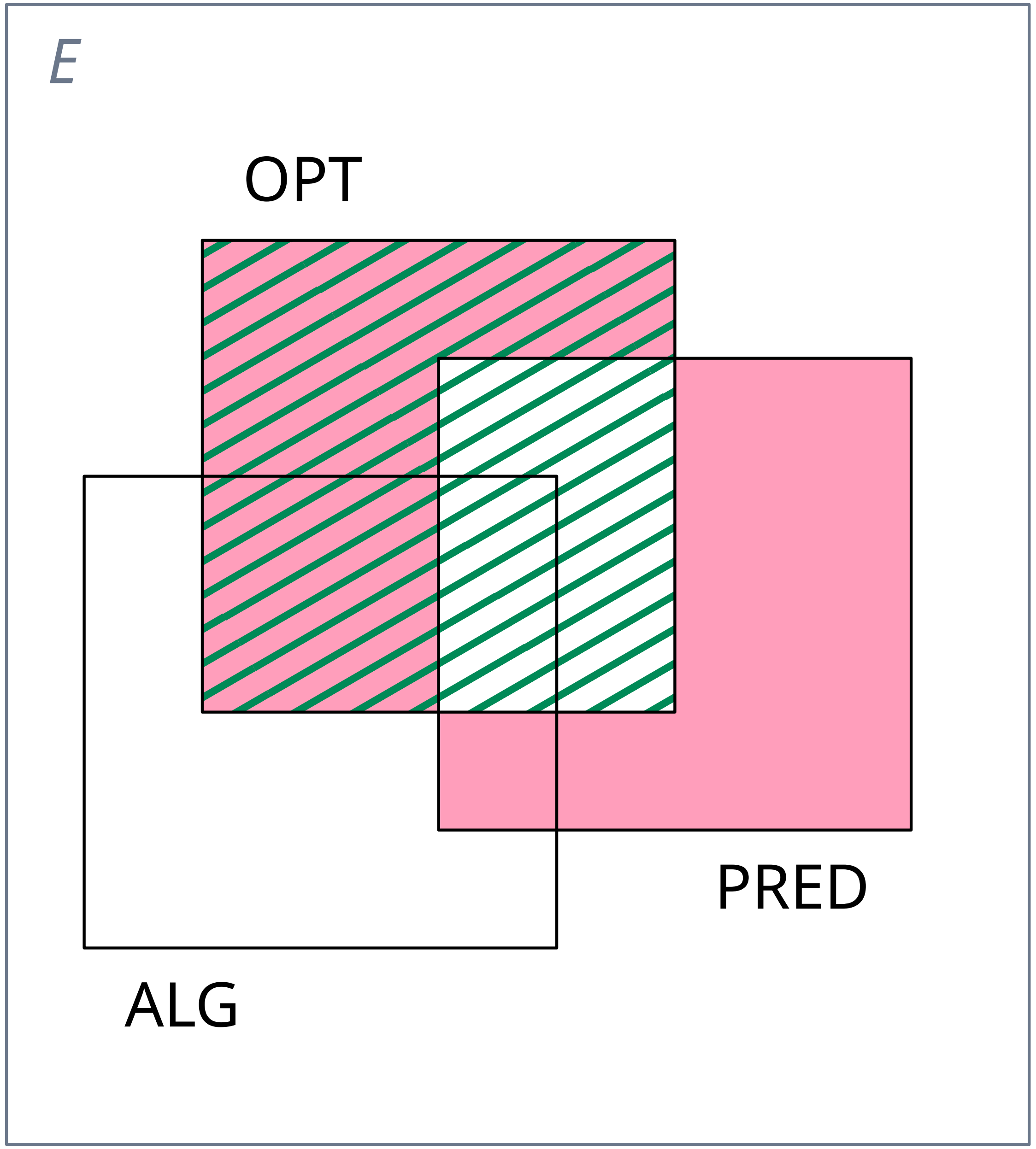


$$\bar{w}(\mathrm{ALG}) \leqslant 2 \cdot \bar{w}(\mathrm{OPT})$$

$$w(\mathrm{ALG} \setminus \mathrm{PRED}) \leqslant 2 \cdot w(\mathrm{OPT} \setminus \mathrm{PRED})$$

$$\begin{aligned} w(\mathrm{ALG}) &\leqslant w(\mathrm{ALG} \cup \mathrm{PRED}) \\ &= w(\mathrm{PRED}) + w(\mathrm{ALG} \setminus \mathrm{PRED}) \\ &\leqslant w(\mathrm{PRED}) + 2 \cdot w(\mathrm{OPT} \setminus \mathrm{PRED}) \\ &\leqslant w(\mathrm{PRED} \cap \mathrm{OPT}) + w(\mathrm{PRED} \setminus \mathrm{OPT}) \\ &\quad + w(\mathrm{OPT} \setminus \mathrm{PRED}) + w(\mathrm{OPT} \setminus \mathrm{PRED}) \end{aligned}$$

The proof



$$\bar{w}(\text{ALG}) \leqslant 2 \cdot \bar{w}(\text{OPT})$$

$$w(\text{ALG} \setminus \text{PRED}) \leqslant 2 \cdot w(\text{OPT} \setminus \text{PRED})$$

$$\begin{aligned} w(\text{ALG}) &\leqslant w(\text{ALG} \cup \text{PRED}) \\ &= w(\text{PRED}) + w(\text{ALG} \setminus \text{PRED}) \\ &\leqslant w(\text{PRED}) + 2 \cdot w(\text{OPT} \setminus \text{PRED}) \\ &\leqslant w(\text{PRED} \cap \text{OPT}) + w(\text{PRED} \setminus \text{OPT}) \\ &\quad + w(\text{OPT} \setminus \text{PRED}) + w(\text{OPT} \setminus \text{PRED}) \\ &= \quad \textbf{OPT} \quad + \quad \boldsymbol{\eta} \qquad \square \end{aligned}$$

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

Output:

- **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** $\min\{\rho, \mathbf{1} + \frac{\eta_+ + (\rho - 1)\cdot\eta_-}{\mathbf{OPT}}\}$

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

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

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]

Applications

- (Minimum Weight) Steiner Tree
- (Minimum Weight) Vertex Cover
- Minimum Weight Perfect Matching in Metric Graphs
- (Maximum Weight) Clique
- Knapsack
- [place for your favorite problem]

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

For Vertex Cover and Clique,
our dependence on $\boldsymbol{\eta}$ is **best possible**
under Unique Games Conjecture

Refined 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

Refined algorithm for Steiner Tree

- choose $\boldsymbol{\alpha} \geqslant 1$
- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \longrightarrow \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \boldsymbol{w}(\boldsymbol{e})/\boldsymbol{\alpha}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned

Refined algorithm for Steiner Tree

- choose $\boldsymbol{\alpha} \geqslant 1$
- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \longrightarrow \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $$\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \boldsymbol{w}(\boldsymbol{e})/\boldsymbol{\alpha}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned

Idea: protect against heavy false positives

Refined algorithm for Steiner Tree

- choose $\boldsymbol{\alpha} \geqslant 1$
- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \longrightarrow \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction **PRED** $\subseteq E$
- create new weight function $\quad \bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \boldsymbol{w}(\boldsymbol{e})/\boldsymbol{\alpha}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned

Idea: protect against heavy false positives

Claim: $w(\mathrm{ALG}) \leqslant (1 + 1/\alpha) \cdot \mathrm{OPT} + \eta_+ + (1 - 1/\alpha) \cdot \eta_-$

Refined algorithm for Steiner Tree

- choose $\boldsymbol{\alpha} \geqslant 1$
- 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 $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \boldsymbol{w}(\boldsymbol{e})/\boldsymbol{\alpha}, & \text{if } \boldsymbol{e} \in \textbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned

Idea: protect against heavy false positives

Claim: $w(\text{ALG}) \leqslant (1 + 1/\alpha) \cdot \text{OPT} + \eta_+ + (1 - 1/\alpha) \cdot \eta_-$

Issue: in terms of $\boldsymbol{\eta}_+, \boldsymbol{\eta}_-$ it only makes sense to choose $\boldsymbol{\alpha} \in \{1, \infty\}$

Refined algorithm for Steiner Tree

- choose $\boldsymbol{\alpha} \geqslant 1$
- read input $\boldsymbol{G} = (V, E)$, $\boldsymbol{w} : E \longrightarrow \mathbb{R}_{\geqslant 0}$, $\boldsymbol{T} \subseteq V$ and prediction $\mathbf{PRED} \subseteq E$
- create new weight function $\bar{\boldsymbol{w}}(\boldsymbol{e}) = \begin{cases} \boldsymbol{w}(\boldsymbol{e})/\boldsymbol{\alpha}, & \text{if } \boldsymbol{e} \in \mathbf{PRED} \\ w(e), & \text{otherwise} \end{cases}$
- run the (near-)linear time **2-approximation** on $(G, \bar{\boldsymbol{w}}, T)$
- return what it returned

Idea: protect against heavy false positives

Claim: $w(\mathrm{ALG}) \leqslant (1 + 1/\alpha) \cdot \mathrm{OPT} + (\alpha - 1) \cdot \sum_{\boldsymbol{p} \in \boldsymbol{P}} \boldsymbol{w}(\boldsymbol{p}) + (1 - 1/\alpha) \cdot \eta_-$

$\boldsymbol{P} := |\mathrm{PRED} \setminus \mathrm{OPT}|$ most expensive paths of MST of T-induced subgraph of metric closure of G

Experiments on Steiner Tree

- Dataset from 2018 PACE Challenge: 200 graphs (with weights and terminals)

Experiments on Steiner Tree

- Dataset from 2018 PACE Challenge: 200 graphs (with weights and terminals)
- Winning solver: CIMAT
  - Iterative (genetic) algorithm
  - Designed to run 30 minutes on each graph
  - Outputs **1.01 · OPT** within **60 seconds** on 95% of graphs

Experiments on Steiner Tree

- Dataset from 2018 PACE Challenge: 200 graphs (with weights and terminals)
- Winning solver: CIMAT
  - Iterative (genetic) algorithm
  - Designed to run 30 minutes on each graph
  - Outputs **1.01 · OPT** within **60 seconds** on 95% of graphs
- Mehlhorn's $O(E + V \log V)$-time 2-approximation algorithm
  - Runs $\leqslant$ **500 milliseconds** on every graph
  - Mean empirical approximation factor: **1.17**

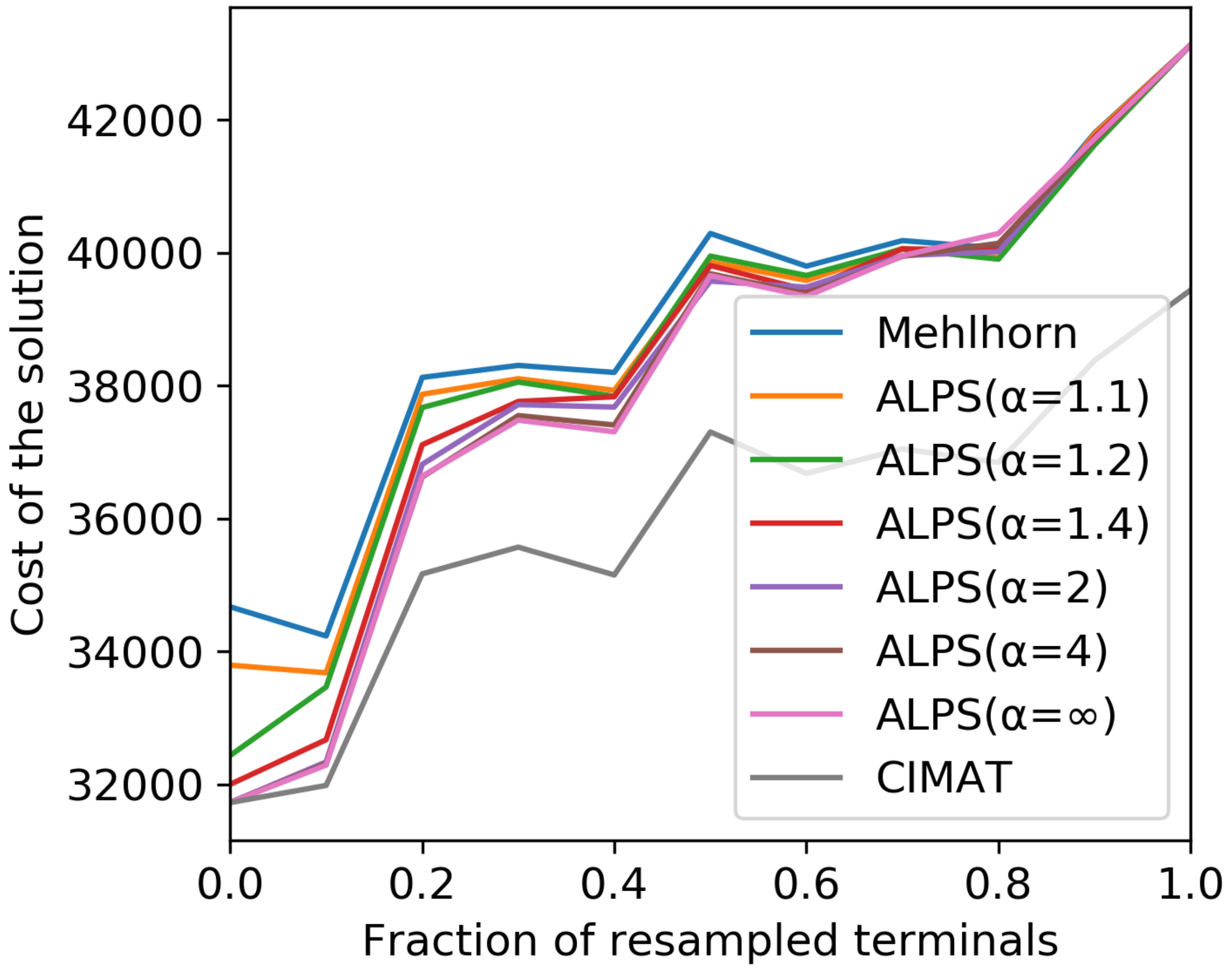
Experiments on Steiner Tree

- Dataset from 2018 PACE Challenge: 200 graphs (with weights and terminals)
- Winning solver: CIMAT
  - Iterative (genetic) algorithm
  - Designed to run 30 minutes on each graph
  - Outputs **1.01 · OPT** within **60 seconds** on 95% of graphs
- Mehlhorn's $O(E + V \log V)$-time 2-approximation algorithm
  - Runs $\leqslant$ **500 milliseconds** on every graph
  - Mean empirical approximation factor: **1.17**
- Each instance (graph, weights, terminals) $\longrightarrow$ distribution over instances
  - Keep the graph and weights fixed
  - Choose $\boldsymbol{p} \in [0, 1]$, resample $\boldsymbol{p}$-fraction of terminals uniformly from all nodes

Experiments on Steiner Tree

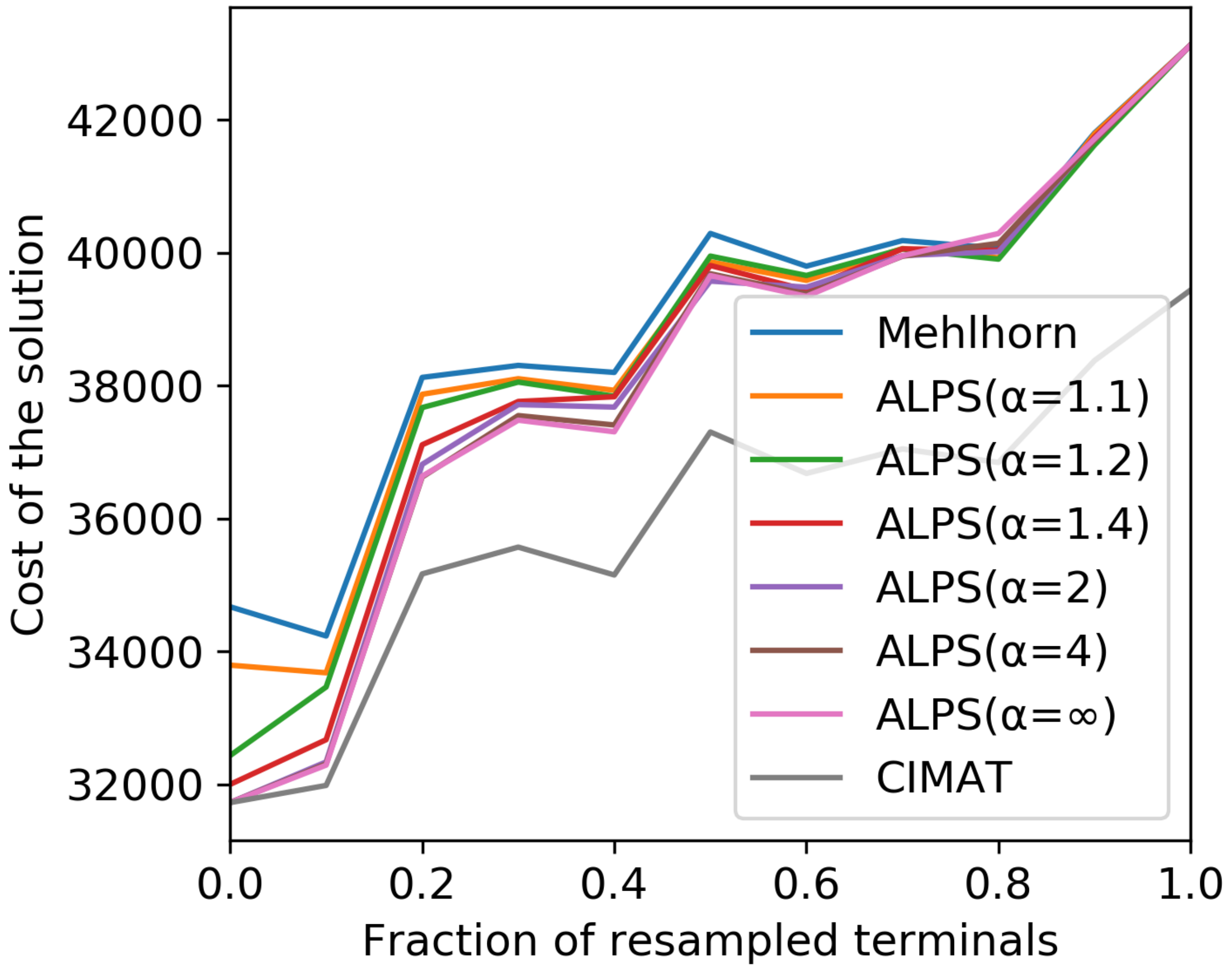
- Dataset from 2018 PACE Challenge: 200 graphs (with weights and terminals)
- Winning solver: CIMAT
  - Iterative (genetic) algorithm
  - Designed to run 30 minutes on each graph
  - Outputs **1.01 · OPT** within **60 seconds** on 95% of graphs
- Mehlhorn's $O(E + V \log V)$-time 2-approximation algorithm
  - Runs $\leqslant$ **500 milliseconds** on every graph
  - Mean empirical approximation factor: **1.17**
- Each instance (graph, weights, terminals) $\longrightarrow$ distribution over instances
  - Keep the graph and weights fixed
  - Choose $\boldsymbol{p} \in [0, 1]$, resample $\boldsymbol{p}$-fraction of terminals uniformly from all nodes
- Leave-one-out cross-validation (10 samples from each distribution)
  - **Learning = empirical risk minimization (edge-wise majority vote)**

Results on individual instances

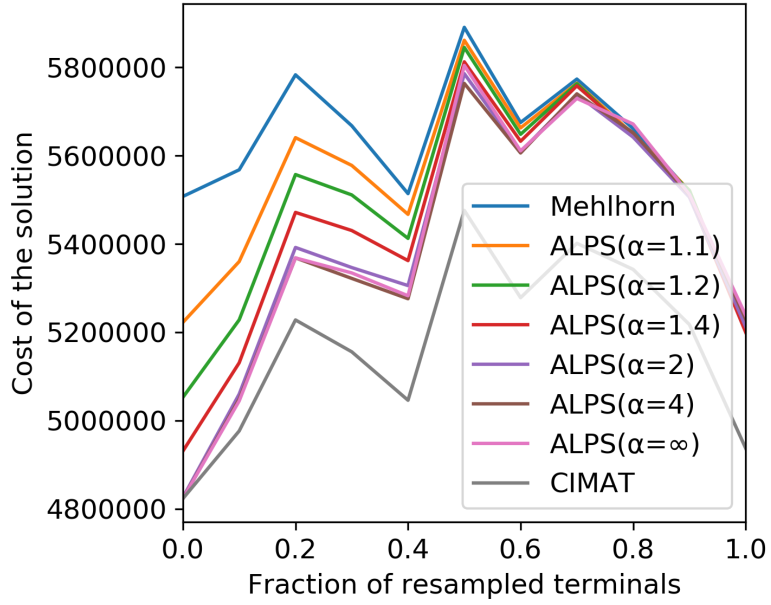


distribution made form graph 001

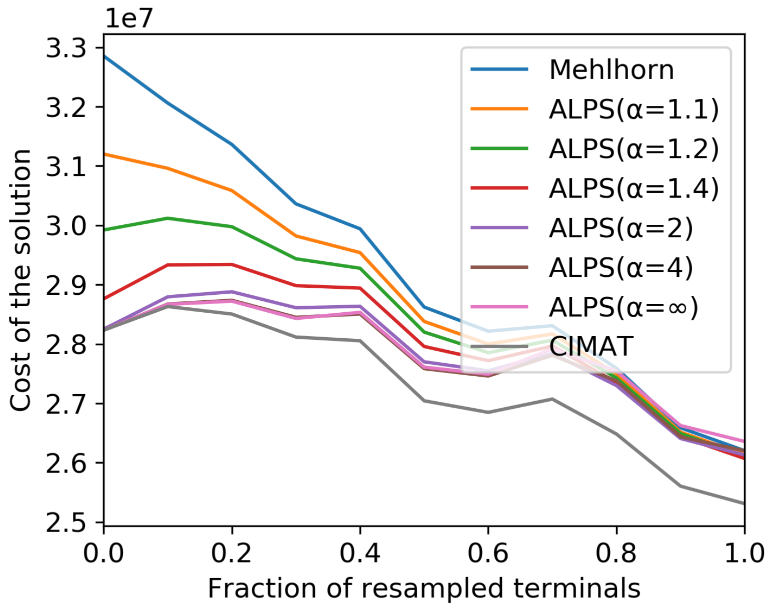
Results on individual instances



distribution made form graph 001

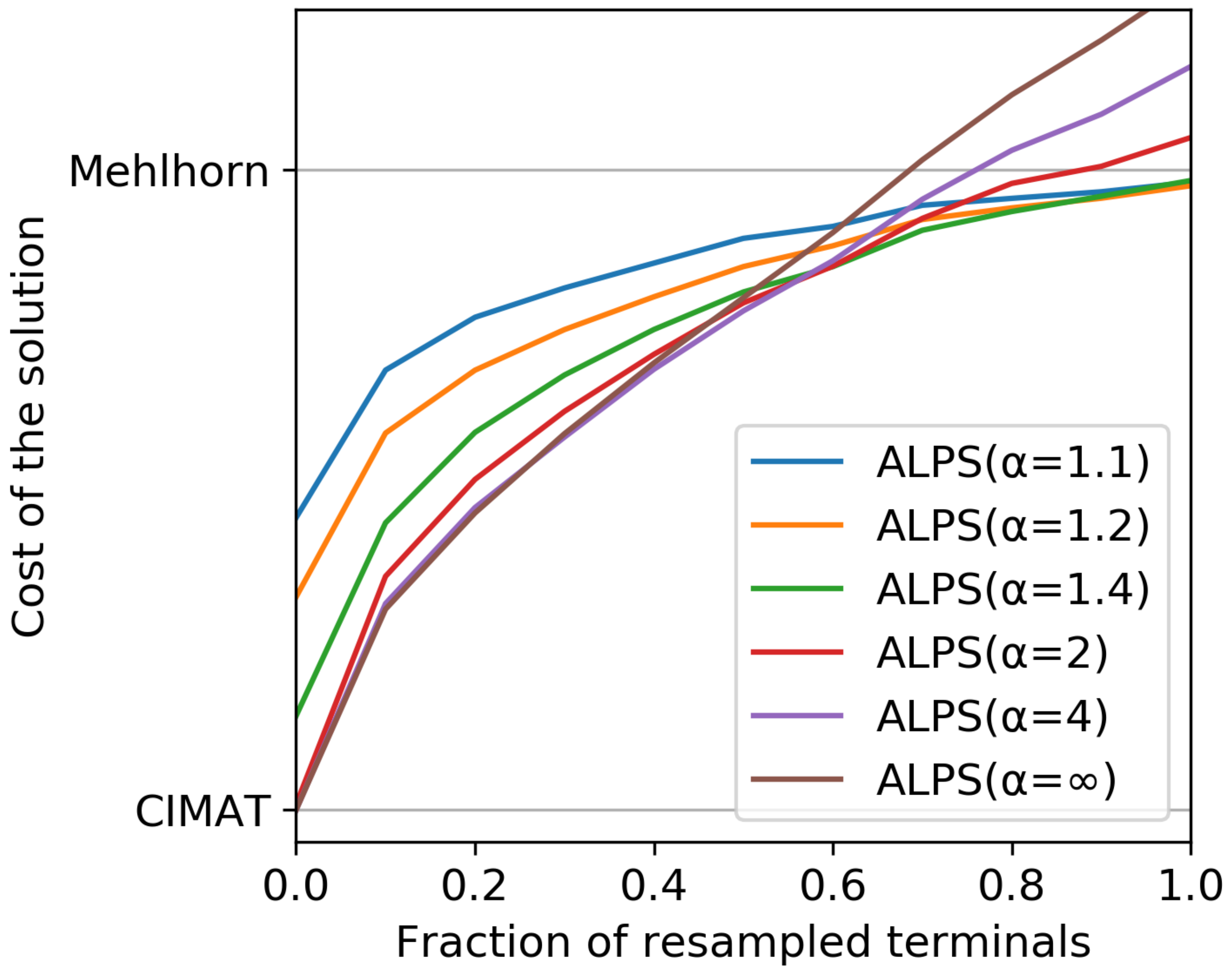


graph 082



graph 178

Results after normalizing and averaging over all instances



Max Cut with ϵ-accurate predictions

[Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi '24]

Input:

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

Output:

- labels $\ell : V \to \{-1, 1\}$ maximizing $\sum_{(u,v) \in E, \ell(u) \neq \ell(v)} \boldsymbol{w(u, v)}$

Max Cut with ϵ-accurate predictions

[Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi '24]

Input:

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

Prediction:

- **labels $\hat{\ell} : V \to \{-1, 1\}$, $\quad \Pr[\hat{\ell}(v) = \ell(v)] = \frac{1}{2} + \varepsilon \quad$ (independently!)**

Output:

- labels $\ell : V \to \{-1, 1\}$ maximizing $\sum_{(u,v) \in E, \ell(u) \neq \ell(v)} w(u, v)$

Max Cut with ϵ-accurate predictions

[Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi '24]

Input:

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

Prediction:

- **labels** $\hat{\ell} : V \to \{-1, 1\}$**,** $\quad \mathbf{Pr}[\hat{\boldsymbol{\ell}}(\boldsymbol{v}) = \boldsymbol{\ell}(\boldsymbol{v})] = \frac{1}{2} + \boldsymbol{\varepsilon}$ $\quad$ **(independently!)**

Output:

- labels $\ell : V \to \{-1, 1\}$ maximizing $\sum_{(\boldsymbol{u},\boldsymbol{v}) \in \boldsymbol{E}, \boldsymbol{\ell}(\boldsymbol{u}) \neq \boldsymbol{\ell}(\boldsymbol{v})} \boldsymbol{w}(\boldsymbol{u}, \boldsymbol{v})$

Result: $(\boldsymbol{\alpha} + \tilde{\boldsymbol{\Omega}}(\boldsymbol{\varepsilon}^4))$-approximation (in polynomial time)

↑ best approximation factor of a classical algorithm ≈ 0.878

Learning-augmented k-means clustering

[Ergun, Feng, Silwal Woodruff, Zhou ’22], [Gamlath, Lattanzi, Norouzi-Fard, Svensson ’22]

Input:

- n points $x_1, x_2, \ldots, x_n \in \mathbb{R}^n$
- number of clusters $k \in \mathbb{Z}_+$

Output:

- labels $\ell : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ minimizing $\boldsymbol{\sum_{i=1}^{n} \left\| x_i - \mathrm{mean}(\{x_j \mid \ell(j) = \ell(i)\}) \right\|^2}$

Learning-augmented k-means clustering

[Ergun, Feng, Silwal Woodruff, Zhou ’22], [Gamlath, Lattanzi, Norouzi-Fard, Svensson ’22]

Input:

- n points $x_1, x_2, \ldots, x_n \in \mathbb{R}^n$
- number of clusters $k \in \mathbb{Z}_+$

Prediction:

- **labels** $\ell : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ consistent with a $\mathbf{(1 + \alpha)}$**-approximate** clustering on $\mathbf{(1 - \lambda)}$**-fraction** of points (per cluster)

Output:

- labels $\ell : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ minimizing $\sum_{i=1}^{n} \left\| x_i - \text{mean}(\{x_j \mid \ell(j) = \ell(i)\}) \right\|^2$

Learning-augmented k-means clustering

[Ergun, Feng, Silwal Woodruff, Zhou '22], [Gamlath, Lattanzi, Norouzi-Fard, Svensson '22]

Input:

- n points $x_1, x_2, \ldots, x_n \in \mathbb{R}^n$
- number of clusters $k \in \mathbb{Z}_+$

Prediction:

- **labels** $\ell : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ consistent with a $\mathbf{(1 + \alpha)}$**-approximate** clustering on $\mathbf{(1 - \lambda)}$**-fraction** of points (per cluster)

Output:

- labels $\ell : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ minimizing $\boldsymbol{\sum_{i=1}^{n} \left\| x_i - \mathrm{mean}(\{x_j \mid \ell(j) = \ell(i)\}) \right\|^2}$

Result: $(\mathbf{1} + \boldsymbol{\emptyset}(\boldsymbol{\alpha}))$-approximation (in polynomial time)

Are **you** solving similar instances of the same problem each day**?**

Are **you** solving similar instances of the same problem each day**?**

Thank you!