Learning-Augmented Online Algorithms for Scheduling and Routing

Nicole Megow

Faculty of Mathematics and Computer Science
University of Bremen

June 2022

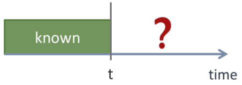
http://schedulingseminar.com

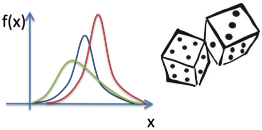
Scheduling under Uncertainty

Scheduling under Uncertainty

Different models for uncertain input

Online Information	Stochastic Information



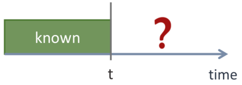


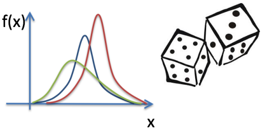
- deterministic uncertainty sets
- explorable uncertainty
- etc.

Scheduling under Uncertainty

Different models for uncertain input

Online Information	Stochastic Information





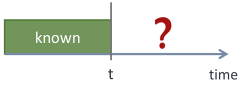
- deterministic uncertainty sets
- explorable uncertainty
- etc.

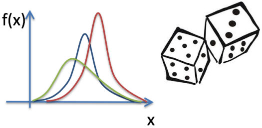
Different optimization frameworks: online optimization, stochastic optimization, robust optimization, explorable uncertainty, etc.

Scheduling under Uncertainty

Different models for uncertain input

Online Information	Stochastic Information	Statistical data or ML predictions



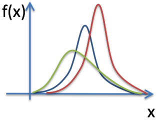


Different optimization frameworks: online optimization, stochastic optimization, robust optimization, explorable uncertainty, etc.

Scheduling under Uncertainty

Different models for uncertain input

Online Information	Stochastic Information	Statistical data or ML predictions



© pixabay

Different optimization frameworks: online optimization, stochastic optimization, robust optimization, explorable uncertainty, etc.

Can error-prone predictions improve upon performance guarantees?

Learning-Augmented Algorithms

- Algorithm with access to prediction
- No assumption on the quality of the prediction

© Adobe Stock

Learning-Augmented Algorithms

- Algorithm with access to prediction
- No assumption on the quality of the prediction

Desired properties

- Consistency: better than worst case if the prediction errors are small
- Robustness: bounded worst-case for arbitrary predictions
- Error-dependency: ideally graceful degradation with the error

Learning-Augmented Algorithms

- Algorithm with access to prediction
- No assumption on the quality of the prediction

Desired properties

- Consistency: better than worst case if the prediction errors are small
- Robustness: bounded worst-case for arbitrary predictions
- Error-dependency: ideally graceful degradation with the error

Line of research initiated by [Lykouris, Vassilvitskii, ICML 2018], and even earlier [Mahidan, Nazerzadeh, Saberi, EC 2007]

Learning-Augmented Algorithms

- Algorithm with access to prediction
- No assumption on the quality of the prediction

Desired properties

- Consistency: better than worst case if the prediction errors are small
- Robustness: bounded worst-case for arbitrary predictions
- Error-dependency: ideally graceful degradation with the error

Line of research initiated by [Lykouris, Vassilvitskii, ICML 2018], and even earlier [Mahidan, Nazerzadeh, Saberi, EC 2007] — became an extremely vibrant area

Learning-Augmented Algorithms

- Algorithm with access to prediction
- No assumption on the quality of the prediction

Desired properties

- Consistency: better than worst case if the prediction errors are small
- Robustness: bounded worst-case for arbitrary predictions
- Error-dependency: ideally graceful degradation with the error

Line of research initiated by [Lykouris, Vassilvitskii, ICML 2018], and even earlier [Mahidan, Nazerzadeh, Saberi, EC 2007] — became an extremely vibrant area

https://algorithms-with-predictions.github.io/

Learning-Augmented Algorithms

- Algorithm with access to prediction
- No assumption on the quality of the prediction

© Adobe Stock

Desired properties

- Consistency: better than worst case if the prediction errors are small
- Robustness: bounded worst-case for arbitrary predictions
- Error-dependency: ideally graceful degradation with the error

Line of research initiated by [Lykouris, Vassilvitskii, ICML 2018], and even earlier [Mahidan, Nazerzadeh, Saberi, EC 2007] — became an extremely vibrant area

https://algorithms-with-predictions.github.io/

Beyond worst case: competitive ratio (online alg.), running time, etc.

Roadmap

Prediction models and learning-augmented algorithms for

1. **Online Scheduling**: uncertain processing times
2. **Online Routing**: uncertain job arrival times and locations

Roadmap

Prediction models and learning-augmented algorithms for

1. **Online Scheduling**: uncertain processing times
2. **Online Routing**: uncertain job arrival times and locations

Important questions

- What to predict? Can we beat worst case bounds?
- How to ensure robustness?
- What is a good error measure?

Online Scheduling

Joint work with Alexander Lindermayr, SPAA 2022.

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine

(Non)-Clairvoyant Scheduling

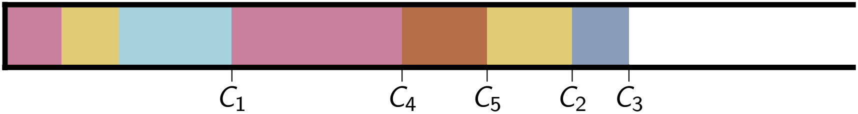
Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine

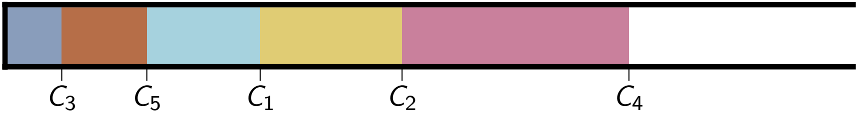


Objective: Minimize sum of completion times $\sum_j C_j$

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine



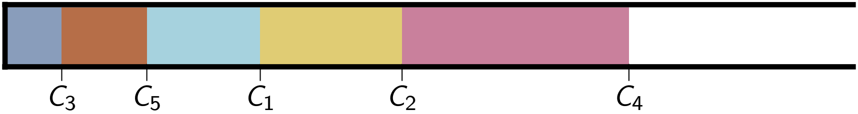
Objective: Minimize sum of completion times $\sum_j C_j$

Optimal Schedule: Shortest Processing Time first (SPT)

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine



Objective: Minimize sum of completion times $\sum_j C_j$ $\qquad \sum_j w_j C_j$

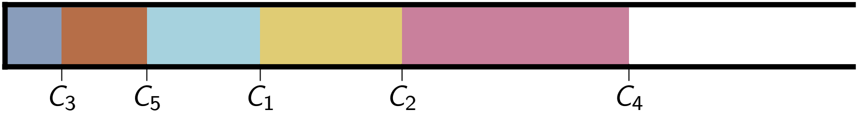
Optimal Schedule: Shortest Processing Time first (SPT)

$\frac{w_1}{p_1} \geq \ldots \geq \frac{w_n}{p_n}$ (WSPT)

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine



Objective: Minimize sum of completion times $\sum_j C_j$ $\sum_j w_j C_j$

Optimal Schedule: Shortest Processing Time first (SPT)

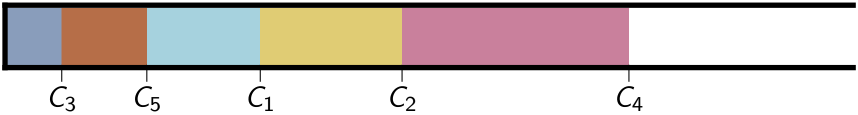
$\frac{w_1}{p_1} \geq \ldots \geq \frac{w_n}{p_n}$ (WSPT)

Processing times must be known (clairvoyant scheduling).

(Non)-Clairvoyant Scheduling

Input: set of jobs with (unknown) processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine



Objective: Minimize sum of completion times $\sum_j C_j$ $\sum_j w_j C_j$

Optimal Schedule: Shortest Processing Time first (SPT)

$\frac{w_1}{p_1} \geq \ldots \geq \frac{w_n}{p_n}$ (WSPT)

Processing times must be known (clairvoyant scheduling).

We assume unknown processing times (non-clairvoyant scheduling).

Non-clairvoyant Scheduling

Competitive analysis (worst-case analysis)

An online algorithm is ρ-competitive if it achieves, for any input instance, a solution of cost within a factor ρ of the optimal cost:

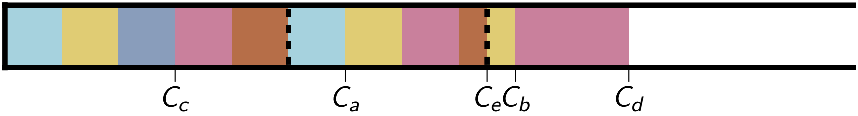
$$\textsc{Alg}(I) \leq \rho \cdot \textsc{Opt}(I), \qquad \text{for any input } I.$$

Non-clairvoyant Scheduling

Round-Robin (RR) is 2-competitive for minimizing $\sum C_j$ on a single machine, and this is best-possible. [Motwani, Phillips, Torng 1994]

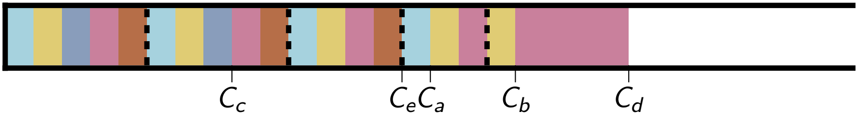
Non-clairvoyant Scheduling

Round-Robin (RR) is 2-competitive for minimizing $\sum C_j$ on a single machine, and this is best-possible. [Motwani, Phillips, Torng 1994]



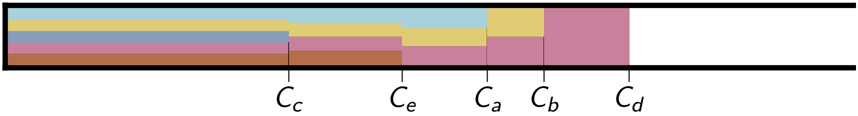
Non-clairvoyant Scheduling

Round-Robin (RR) is 2-competitive for minimizing $\sum C_j$ on a single machine, and this is best-possible. [Motwani, Phillips, Torng 1994]



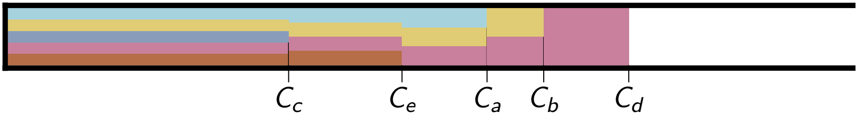
Non-clairvoyant Scheduling

Round-Robin (RR) is 2-competitive for minimizing $\sum C_j$ on a single machine, and this is best-possible. [Motwani, Phillips, Torng 1994]



Non-clairvoyant Scheduling

Round-Robin (RR) is 2-competitive for minimizing $\sum C_j$ on a single machine, and this is best-possible. [Motwani, Phillips, Torng 1994]



Further **Time-Sharing** algorithms for more general problems:

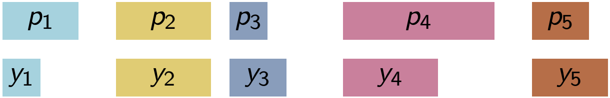
- Individual job weights: Weighted Round-Robin (2-competitive) [Kim, Chwa 2003]
- Identical machines: Weighted Dynamic Equipartition (2-comp.) [Beaumont, Bonichon, Eyraud-Dubois, Marchal 2012]
- Unrelated machines: Proportional Fairness (128-competitive) [Im, Kulkarni, Munagala 2018]

Length Predictions and ℓ_1-errors

Length Predictions and ℓ_1-errors

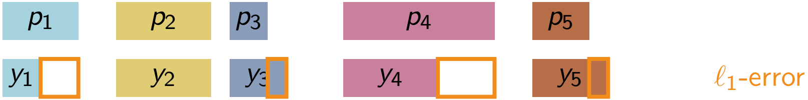
Predict job lengths y_j

[Kumar, Purohit, Svitkina 2018]
[Wei, Zhang 2020], [Im, Kumar, Qaem, Purohit 2021]



Length Predictions and ℓ_1-errors

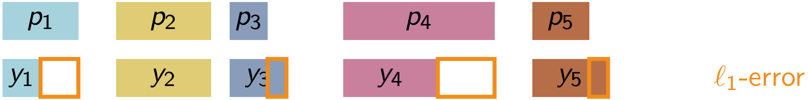
Predict job lengths y_j [Kumar, Purohit, Svitkina 2018]
[Wei, Zhang 2020], [Im, Kumar, Qaem, Purohit 2021]



Error: $\ell_1 = \sum_{j=1}^{n} |p_j - y_j|$

Length Predictions and ℓ_1-errors

Predict job lengths y_j [Kumar, Purohit, Svitkina 2018]
[Wei, Zhang 2020], [Im, Kumar, Qaem, Purohit 2021]

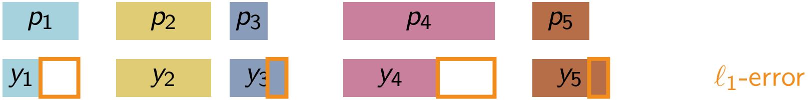


Error: $\ell_1 = \sum_{j=1}^{n} |p_j - y_j|$

Natural algorithm: run shortest predicted job first (SPF).
("Follow the prediction": SPT on y_j)

Length Predictions and ℓ_1-errors

Predict job lengths y_j [Kumar, Purohit, Svitkina 2018]
[Wei, Zhang 2020], [Im, Kumar, Qaem, Purohit 2021]



Error: $\ell_1 = \sum_{j=1}^{n} |p_j - y_j|$

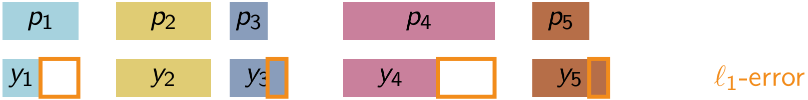
Natural algorithm: run shortest predicted job first (SPF).
("Follow the prediction": SPT on y_j)

Lemma [Kumar, Purohit, Svitkina 2018]

SPF achieves scheduling cost $\mathrm{SPF}(y_j, p_j) \leq \textsc{Opt}(p_j) + n \cdot \ell_1$.

Length Predictions and ℓ_1-errors

Predict job lengths y_j [Kumar, Purohit, Svitkina 2018]
[Wei, Zhang 2020], [Im, Kumar, Qaem, Purohit 2021]



Error: $\ell_1 = \sum_{j=1}^{n} |p_j - y_j|$

Natural algorithm: run shortest predicted job first (SPF).
("Follow the prediction": SPT on y_j)

Lemma [Kumar, Purohit, Svitkina 2018]

SPF achieves scheduling cost $\mathrm{SPF}(y_j, p_j) \leq \textsc{Opt}(p_j) + n \cdot \ell_1$.

Consistent but not robust (against bad predictions).

Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

Input:

- prediction-clairvoyant alg. $\mathcal{A}^C$ ("follow the prediction") with some error-dependent competitive ratio
- non-clairvoyant alg. $\mathcal{A}^N$ with error-independent competitive ratio
- confidence parameter $\lambda \in (0, 1)$

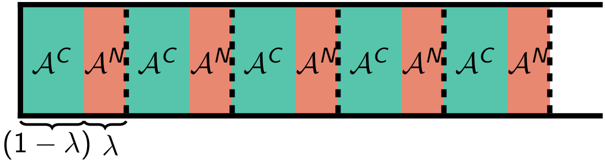
Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

Input:

- prediction-clairvoyant alg. $\mathcal{A}^C$ ("follow the prediction") with some error-dependent competitive ratio
- non-clairvoyant alg. $\mathcal{A}^N$ with error-independent competitive ratio
- confidence parameter $\lambda \in (0,1)$

Preferential Time Sharing ($\lambda, \mathcal{A}^C, \mathcal{A}^N$)



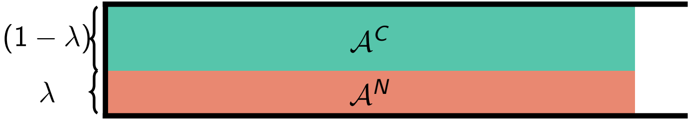
Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

Input:

- prediction-clairvoyant alg. $\mathcal{A}^C$ ("follow the prediction") with some error-dependent competitive ratio
- non-clairvoyant alg. $\mathcal{A}^N$ with error-independent competitive ratio
- confidence parameter $\lambda \in (0, 1)$

Preferential Time Sharing $(\lambda, \mathcal{A}^C, \mathcal{A}^N)$



Motivation: $\mathcal{A}^C$ gives consistency, $\mathcal{A}^N$ gives robustness, trade-off by λ

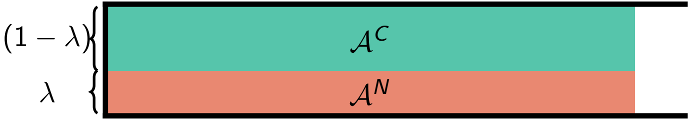
Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

Input:

- prediction-clairvoyant alg. $\mathcal{A}^C$ ("follow the prediction") with some error-dependent competitive ratio
- non-clairvoyant alg. $\mathcal{A}^N$ with error-independent competitive ratio
- confidence parameter $\lambda \in (0, 1)$

Preferential Time Sharing ($\lambda, \mathcal{A}^C, \mathcal{A}^N$)



Motivation: $\mathcal{A}^C$ gives consistency, $\mathcal{A}^N$ gives robustness, trade-off by λ

Monotone algorithms: if p_j's shrink, completion times do not increase

Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

Theorem

PTS$(\lambda, \mathcal{A}^C, \mathcal{A}^N)$ has competitive ratio min $\left\{\frac{1}{1-\lambda}\left(\alpha + \frac{\eta}{\mathrm{OPT}}\right), \frac{\beta}{\lambda}\right\}$, if

- $\mathcal{A}^C$ is monotone and $\left(\alpha + \frac{\eta}{\mathrm{OPT}}\right)$-competitive and
- $\mathcal{A}^N$ is monotone and β-competitive.

Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

Theorem

PTS($\lambda, \mathcal{A}^C, \mathcal{A}^N$) has competitive ratio $\min\left\{\frac{1}{1-\lambda}\left(\alpha + \frac{\eta}{\mathrm{OPT}}\right), \frac{\beta}{\lambda}\right\}$, if

- $\mathcal{A}^C$ is monotone and $\left(\alpha + \frac{\eta}{\mathrm{OPT}}\right)$-competitive and
- $\mathcal{A}^N$ is monotone and β-competitive.

Corollary. PTS(λ, SPF, RR) achieves for non-clairvoyant $1|pmtn|\sum C_j$ with length predictions, for any $\lambda \in (0,1)$, a competitive ratio of

$$\min\left\{\frac{1}{1-\lambda}\left(1 + \frac{n \cdot \ell_1}{\mathrm{OPT}}\right), \frac{2}{\lambda}\right\}.$$

Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018]

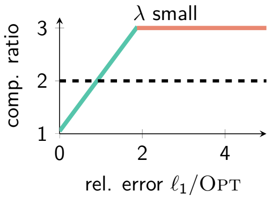
Theorem

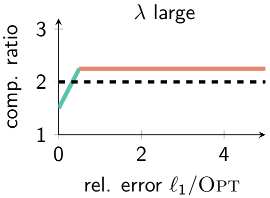
PTS($\lambda, \mathcal{A}^C, \mathcal{A}^N$) has competitive ratio $\min\left\{\frac{1}{1-\lambda}\left(\alpha + \frac{\eta}{\textsc{Opt}}\right), \frac{\beta}{\lambda}\right\}$, if

- $\mathcal{A}^C$ is monotone and $\left(\alpha + \frac{\eta}{\textsc{Opt}}\right)$-competitive and
- $\mathcal{A}^N$ is monotone and β-competitive.

Corollary. PTS(λ, SPF, RR) achieves for non-clairvoyant $1|pmtn|\sum C_j$ with length predictions, for any $\lambda \in (0,1)$, a competitive ratio of

$$\min\left\{\frac{1}{1-\lambda}\left(1 + \frac{n \cdot \ell_1}{\textsc{Opt}}\right), \frac{2}{\lambda}\right\}.$$





Can the PTS framework be generalized?

Can the PTS framework be generalized?

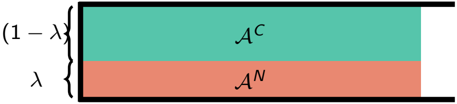
(Informal) Theorem [Lindermayr and M., 2022]

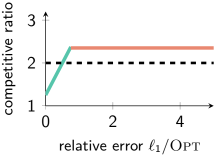
The PTS theorem can be generalized to scheduling settings with unrelated machines, release dates and weights.

Can the PTS framework be generalized?

(Informal) Theorem [Lindermayr and M., 2022]

The PTS theorem can be generalized to scheduling settings with unrelated machines, release dates and weights.

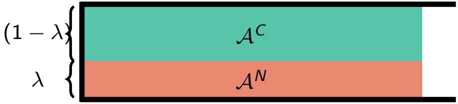


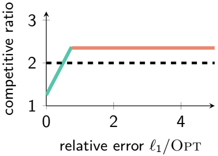


Can the PTS framework be generalized?

(Informal) Theorem [Lindermayr and M., 2022]

The PTS theorem can be generalized to scheduling settings with unrelated machines, release dates and weights.





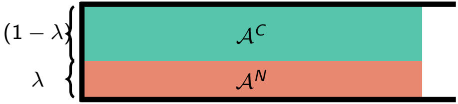
Roadmap

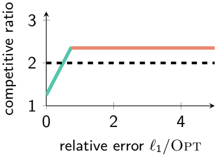
1. develop monotone prediction-clairvoyant alg. $\mathcal{A}^C$ and error-dependent competitive ratio
2. select a monotone non-clairvoyant algorithm $\mathcal{A}^N$

Can the PTS framework be generalized?

(Informal) Theorem [Lindermayr and M., 2022]

The PTS theorem can be generalized to scheduling settings with unrelated machines, release dates and weights.





Roadmap

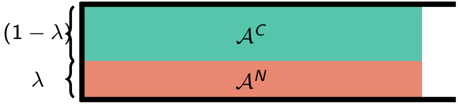
1. develop monotone prediction-clairvoyant alg. $\mathcal{A}^C$ and error-dependent competitive ratio
2. select a monotone non-clairvoyant algorithm $\mathcal{A}^N$

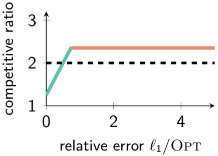
Obstacle: Proving error-dependent bounds seems difficult with ℓ_1-error (linear error vs. quadratic objective)

Can the PTS framework be generalized?

(Informal) Theorem [Lindermayr and M., 2022]

The PTS theorem can be generalized to scheduling settings with unrelated machines, release dates and weights.





Roadmap

1. develop monotone prediction-clairvoyant alg. $\mathcal{A}^C$ and error-dependent competitive ratio
2. select a monotone non-clairvoyant algorithm $\mathcal{A}^N$

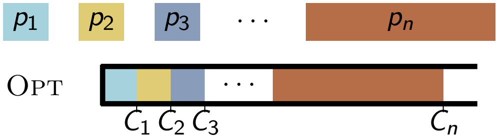
Obstacle: Proving error-dependent bounds seems difficult with ℓ_1-error (linear error vs. quadratic objective) $\longrightarrow$ **What is a "good" error measure?**

Discussion ℓ_1-Error Measure

ℓ_1-error cannot distinguish well between "good" and "bad" predictions

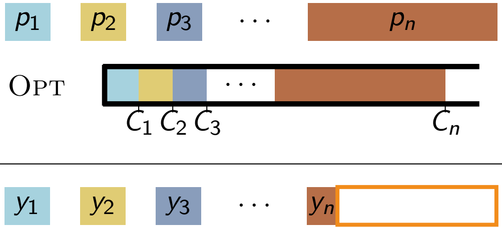
Discussion ℓ_1-Error Measure

ℓ_1-error cannot distinguish well between "good" and "bad" predictions



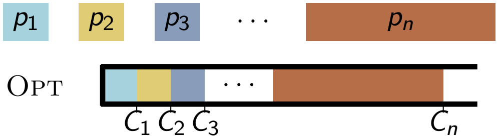
Discussion ℓ_1-Error Measure

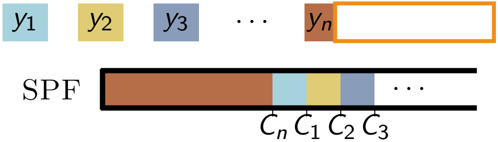
ℓ_1-error cannot distinguish well between "good" and "bad" predictions



Discussion ℓ_1-Error Measure

ℓ_1-error cannot distinguish well between "good" and "bad" predictions



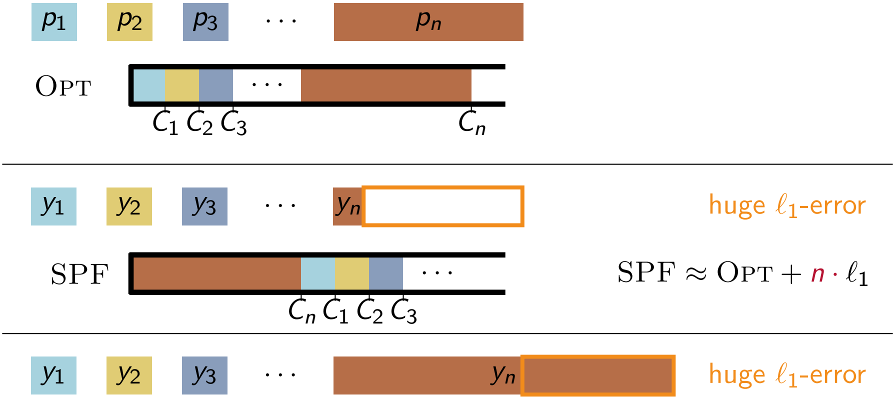


huge ℓ_1-error

$\mathrm{SPF} \approx \textsc{Opt} + n \cdot \ell_1$

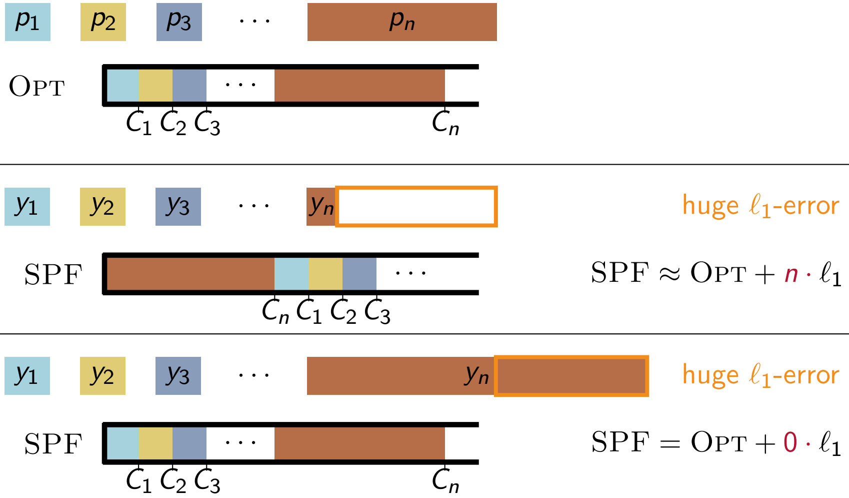
Discussion ℓ_1-Error Measure

ℓ_1-error cannot distinguish well between "good" and "bad" predictions



Discussion ℓ_1-Error Measure

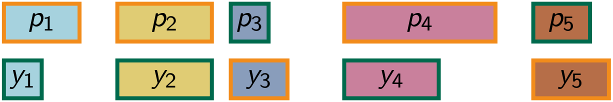
ℓ_1-error cannot distinguish well between "good" and "bad" predictions



Length Predictions and ν-error

[Im, Kumar, Qaem, Purohit 2021]

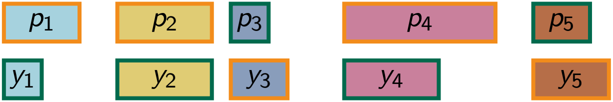
Alternative error: $\nu = \textsc{Opt}(\{\max\{p_j, y_j\}\}_j) - \textsc{Opt}(\{\min\{p_j, y_j\}\}_j)$



Length Predictions and ν-error

[Im, Kumar, Qaem, Purohit 2021]

Alternative error: $\nu = \textsc{Opt}(\{\max\{p_j, y_j\}\}_j) - \textsc{Opt}(\{\min\{p_j, y_j\}\}_j)$

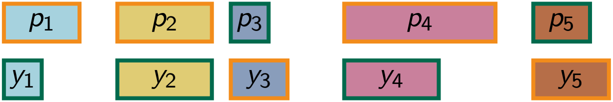


Error-tracker algorithm: $\min\left\{(1+\epsilon)\textsc{Opt} + \mathcal{O}\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu, \frac{2}{\epsilon}\textsc{Opt}\right\}$ for $1|pmtn|\sum C_j$

Length Predictions and ν-error

[Im, Kumar, Qaem, Purohit 2021]

Alternative error: $\nu = \textsc{Opt}(\{\max\{p_j, y_j\}\}_j) - \textsc{Opt}(\{\min\{p_j, y_j\}\}_j)$

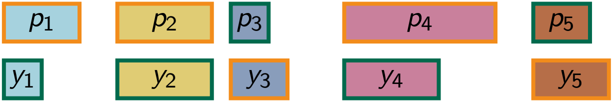


Error-tracker algorithm: $\min\left\{(1+\epsilon)\textsc{Opt} + \mathcal{O}\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu,\ \frac{2}{\epsilon}\textsc{Opt}\right\}$
for $1|pmtn|\sum C_j$ (seems challenging to generalize)

Length Predictions and ν-error

[Im, Kumar, Qaem, Purohit 2021]

Alternative error: $\nu = \textsc{Opt}(\{\max\{p_j, y_j\}\}_j) - \textsc{Opt}(\{\min\{p_j, y_j\}\}_j)$



Error-tracker algorithm: $\min\left\{(1+\epsilon)\textsc{Opt} + \mathcal{O}\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu,\ \frac{2}{\epsilon}\textsc{Opt}\right\}$
for $1|pmtn|\sum C_j$ (seems challenging to generalize)

Issue: Underestimates true difficulty of the following instance

$p_1 = y_1 = \ldots = p_{n-1} = y_{n-1} = 1$, but $p_n = n^2$ and $y_n = 0$.

$\nu = n^2 + n = \Theta(n^2)$ vs $\mathrm{SPF}(y_j, p_j) - \textsc{Opt}(p_j) = \Omega(n^3)$

Permutation Predictions

Permutation predictions: predict an order of jobs: $\hat{\sigma} : [n] \to [n]$

Motivation: knowing WSPT order is often sufficient for good approximations:

- optimal for $1|(pmtn)|\sum w_j C_j$ [Smith 1956]
- 2-competitive for $P|r_j, pmtn|\sum w_j C_j$ [**M**. & Schulz 2004]
- 5.83-competitive for $R|r_j, pmtn|\sum w_j C_j$ [Lindermayr & **M**. 2022]

Permutation Predictions

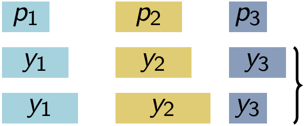
Permutation predictions: predict an order of jobs: $\hat{\sigma} : [n] \to [n]$

p_1 p_2 p_3 (W)SPT order $p_3 \leq p_1 \leq p_2$

$$\left.\begin{array}{lll} y_1 & y_2 & y_3 \\ y_1 & y_2 & y_3 \end{array}\right\}\text{Indicate correct order } y_3 \leq y_1 \leq y_2\text{, but } \ell_1, \nu > 0$$

Permutation Predictions

Permutation predictions: predict an order of jobs: $\hat{\sigma} : [n] \to [n]$



(W)SPT order $p_3 \leq p_1 \leq p_2$

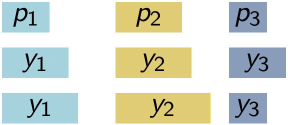
Indicate correct order $y_3 \leq y_1 \leq y_2$, but $\ell_1, \nu > 0$

Error measure: quantifies effect of inversions $\mathcal{I}$ between $\hat{\sigma}$ and true WSPT order on list scheduling according to predicted order:

$$\eta^S = \sum_{(i,j) \in \mathcal{I}} (w_i p_j - w_j p_i)$$

Permutation Predictions

Permutation predictions: predict an order of jobs: $\hat{\sigma} : [n] \to [n]$



(W)SPT order $p_3 \le p_1 \le p_2$

Indicate correct order $y_3 \le y_1 \le y_2$, but $\ell_1, \nu > 0$

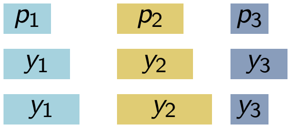
Error measure: quantifies effect of inversions $\mathcal{I}$ between $\hat{\sigma}$ and true WSPT order on list scheduling according to predicted order:

$$\eta^S = \sum_{(i,j)\in\mathcal{I}} (w_i p_j - w_j p_i)$$

- For $1||\sum w_j C_j$ this is exactly $\eta^S = \textsc{Opt}(\hat{\sigma}) - \textsc{Opt}(\sigma)$.

Permutation Predictions

Permutation predictions: predict an order of jobs: $\hat{\sigma} : [n] \to [n]$



(W)SPT order $p_3 \leq p_1 \leq p_2$

Indicate correct order $y_3 \leq y_1 \leq y_2$, but $\ell_1, \nu > 0$

Error measure: quantifies effect of inversions $\mathcal{I}$ between $\hat{\sigma}$ and true WSPT order on list scheduling according to predicted order:

$$\eta^S = \sum_{(i,j)\in\mathcal{I}} (w_i p_j - w_j p_i)$$

- For $1||\sum w_j C_j$ this is exactly $\eta^S = \textsc{Opt}(\hat{\sigma}) - \textsc{Opt}(\sigma)$.
- η^S captures structure instead of irrelevant numerical values.

Permutation Predictions

Permutation predictions: predict an order of jobs: $\hat{\sigma} : [n] \to [n]$

p_1 p_2 p_3 (W)SPT order $p_3 \leq p_1 \leq p_2$

y_1 y_2 y_3

y_1 y_2 y_3

Indicate correct order $y_3 \leq y_1 \leq y_2$, but $\ell_1, \nu > 0$

Error measure: quantifies effect of inversions $\mathcal{I}$ between $\hat{\sigma}$ and true WSPT order on list scheduling according to predicted order:

$$\eta^S = \sum_{(i,j) \in \mathcal{I}} (w_i p_j - w_j p_i)$$

- For $1||\sum w_j C_j$ this is exactly $\eta^S = \text{OPT}(\hat{\sigma}) - \text{OPT}(\sigma)$.
- η^S captures structure instead of irrelevant numerical values.
- Permutation predictions are efficiently PAC-learnable w.r.t. η^S.

PTS and Permutation Predictions (1)

PTS for weighted jobs on a single machine $1|pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: WSPT (monotone optimal) [Smith 1956]

Schedule jobs in WSPT order

$$\min\left\{ \frac{1}{1-\lambda} \cdot \left(\qquad\qquad \right), \frac{1}{\lambda} \cdot \quad \right\}$$

PTS and Permutation Predictions (1)

PTS for weighted jobs on a single machine $1|pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: WSPT (monotone optimal) [Smith 1956]

Schedule jobs in **predicted** order $\hat{\sigma}$

$$\min\left\{\frac{1}{1-\lambda}\cdot\left(1+\frac{\eta^S}{\textsc{Opt}}\right),\frac{1}{\lambda}\cdot\quad\right\}$$

PTS and Permutation Predictions (1)

PTS for weighted jobs on a single machine $1|pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: WSPT (monotone optimal) [Smith 1956]

Schedule jobs in **predicted** order $\hat{\sigma}$

- non-clairvoyant $\mathcal{A}^N$: WRR (monotone 2-competitive) [Kim, Chwa 2003]

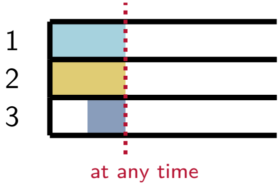
$$\min\left\{\frac{1}{1-\lambda}\cdot\left(1+\frac{\eta^S}{\textsc{Opt}}\right),\frac{1}{\lambda}\cdot 2\right\}$$

PTS and Permutation Predictions (2)

PTS for m identical machines and release dates $P|r_j, pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (monotone 2-competitive)

[**M**. and Schulz 2004]



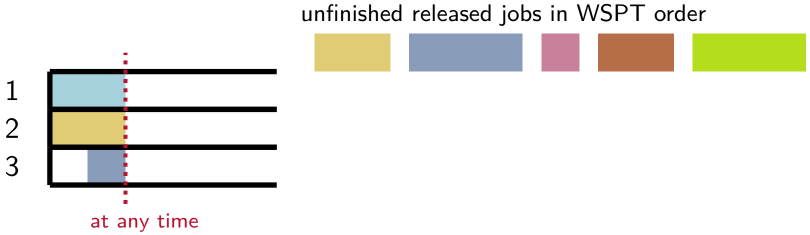
$$\min \left\{ \frac{1}{1-\lambda} \cdot \left(\qquad \qquad \right), \frac{1}{\lambda} \cdot \quad \right\}$$

PTS and Permutation Predictions (2)

PTS for m identical machines and release dates $P|r_j, pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (monotone 2-competitive)

[**M**. and Schulz 2004]



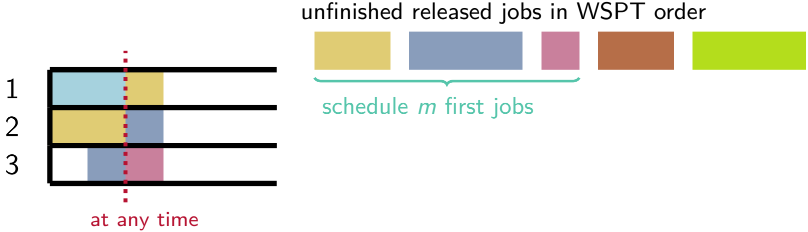
$$\min\left\{\frac{1}{1-\lambda}\cdot\left(\qquad\qquad \right), \frac{1}{\lambda}\cdot \quad \right\}$$

PTS and Permutation Predictions (2)

PTS for m identical machines and release dates $P|r_j, pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (monotone 2-competitive)

[**M**. and Schulz 2004]



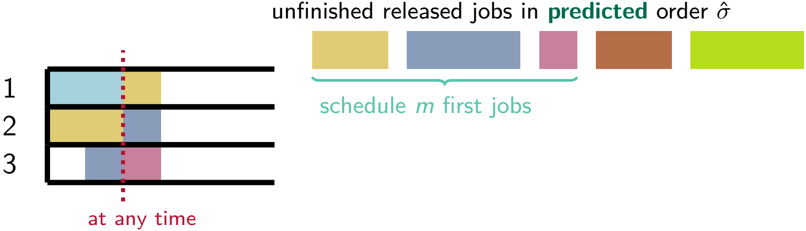
$$\min\left\{\frac{1}{1-\lambda}\cdot\left(\qquad\qquad\right), \frac{1}{\lambda}\cdot\quad\right\}$$

PTS and Permutation Predictions (2)

PTS for m identical machines and release dates $P|r_j, pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (monotone 2-competitive)

[**M**. and Schulz 2004]



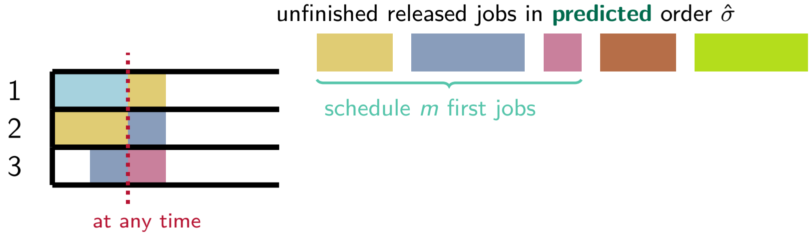
$$\min\left\{\frac{1}{1-\lambda}\cdot\left(2+\frac{\eta^S}{m\cdot\text{OPT}}\right),\frac{1}{\lambda}\cdot\quad\right\}$$

PTS and Permutation Predictions (2)

PTS for m identical machines and release dates $P|r_j, pmtn|\sum w_j C_j$

- prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (monotone 2-competitive)

[**M**. and Schulz 2004]



- non-clairvoyant $\mathcal{A}^N$: WDEQ (monotone 3-competitive)

[Beaumont, Bonichon, Eyraud-Dubois and Marchal 2012]

$$\min\left\{\frac{1}{1-\lambda}\cdot\left(2+\frac{\eta^S}{m\cdot\textsc{Opt}}\right), \frac{1}{\lambda}\cdot 3\right\}$$

PTS and Permutation Predictions (3)

PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$

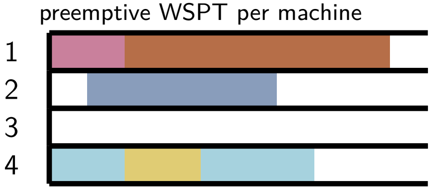
PTS and Permutation Predictions (3)

PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \dots, \hat{\sigma}_m$

- prediction-clairvoyant $\mathcal{A}^C$: MinIncrease+WSPT (mon. 5.83-comp.)

[Lindermayr and **M**. 2022]



$$\min\left\{ \frac{1}{1-\lambda} \cdot \left(\qquad \right), \frac{1}{\lambda} \cdot \qquad \right\}$$

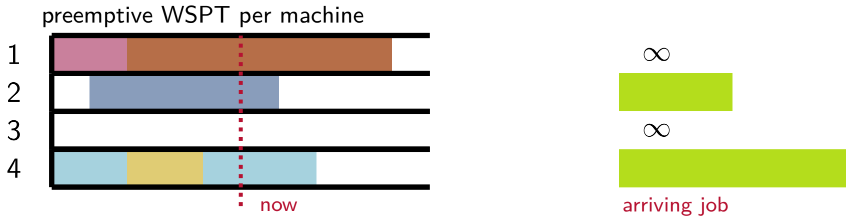
PTS and Permutation Predictions (3)

PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$

- prediction-clairvoyant $\mathcal{A}^C$: MinIncrease+WSPT (mon. 5.83-comp.)

[Lindermayr and **M**. 2022]



$$\min\left\{\frac{1}{1-\lambda}\cdot\left(\qquad\qquad\qquad \right), \frac{1}{\lambda}\cdot \qquad \right\}$$

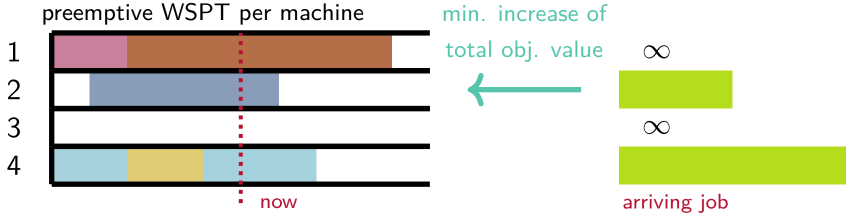
PTS and Permutation Predictions (3)

PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$

- prediction-clairvoyant $\mathcal{A}^C$: MinIncrease+WSPT (mon. 5.83-comp.)

[Lindermayr and **M.** 2022]



$$\min\left\{\frac{1}{1-\lambda}\cdot\left(\qquad\qquad\qquad\right), \frac{1}{\lambda}\cdot\qquad\right\}$$

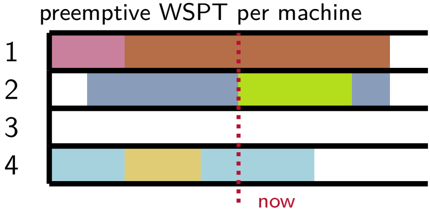
PTS and Permutation Predictions (3)

PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$

- prediction-clairvoyant $\mathcal{A}^C$: MinIncrease+WSPT (mon. 5.83-comp.)

[Lindermayr and **M**. 2022]



$$\min \left\{ \frac{1}{1-\lambda} \cdot \left(5.8284 + \frac{\eta^R}{\textsc{Opt}} \right), \frac{1}{\lambda} \cdot \qquad \right\}$$

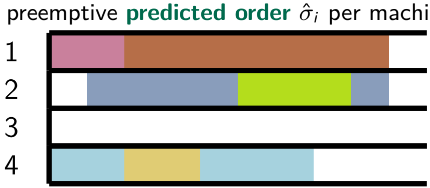
PTS and Permutation Predictions (3)

PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$

- prediction-clairvoyant $\mathcal{A}^C$: MinIncrease+WSPT (mon. 5.83-comp.)

[Lindermayr and **M.** 2022]



assign job j to

predicted machine i

i.e. $i \in [m]$ s.t. $j \in \hat{\sigma}_i$

$$\min\left\{\frac{1}{1-\lambda}\cdot\left(5.8284+\frac{\eta^R}{\textsc{Opt}}\right), \frac{1}{\lambda}\cdot \qquad \right\}$$

PTS and Permutation Predictions (3)

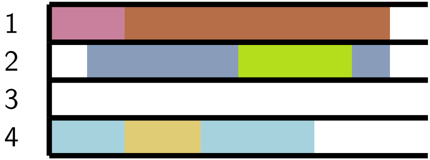
PTS on m unrelated machines $R|r_j, pmtn|\sum w_j C_j$:

Predict machine allocation and permutation for each machine $\hat{\sigma}_1, \ldots, \hat{\sigma}_m$

- prediction-clairvoyant $\mathcal{A}^C$: MinIncrease+WSPT (mon. 5.83-comp.)

[Lindermayr and **M.** 2022]

preemptive **predicted order** $\hat{\sigma}_i$ per machine i



assign job j to

predicted machine i

i.e. $i \in [m]$ s.t. $j \in \hat{\sigma}_i$

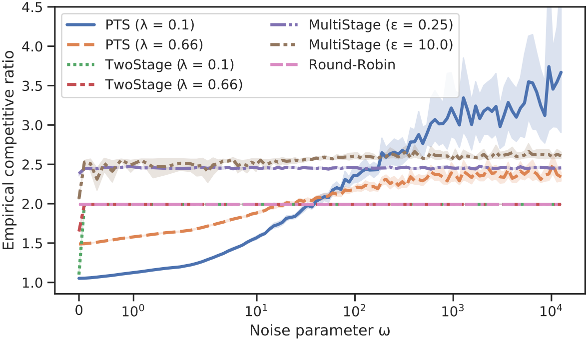
- non-clairvoyant $\mathcal{A}^N$: Proportional Fairness (mon. 128-competitive)

[Im, Kulkarni and Munagala 2018]

$$\min \left\{ \frac{1}{1-\lambda} \cdot \left(5.8284 + \frac{\eta^R}{\textsc{Opt}} \right), \frac{1}{\lambda} \cdot 128 \right\}$$

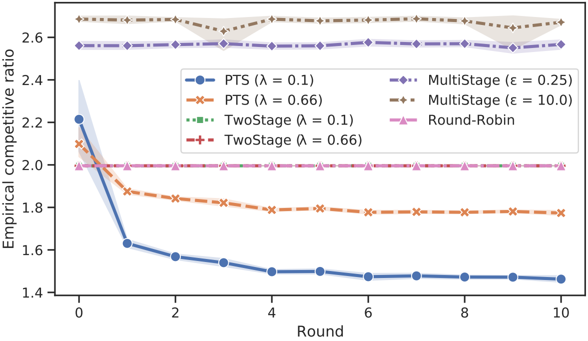
Sensitivity Experiments

- Single machine, unweighted jobs (PTS equals algorithm in [KPS18])
- Synthetic instances sampled from Pareto-distribution with shape 1.1
  Many small jobs and few very large jobs!



Online Learning Experiments

- learn prediction from previous instances
- already after one round performance improves substantially
- such instances can be learned fast in practice (detect long jobs)



Intermediate Summary

Non-clairvoyant scheduling with predictions

- Discussion of length and permutation prediction and error measures

Intermediate Summary

Non-clairvoyant scheduling with predictions

- Discussion of length and permutation prediction and error measures
- Powerful algorithmic framework that admits (blackbox) algorithms with certain properties (error-dependent comp. ratio, monotone)

Intermediate Summary

Non-clairvoyant scheduling with predictions

- Discussion of length and permutation prediction and error measures
- Powerful algorithmic framework that admits (blackbox) algorithms with certain properties (error-dependent comp. ratio, monotone)
- First results for weights, release dates, multiple machine models

Intermediate Summary

Non-clairvoyant scheduling with predictions

- Discussion of length and permutation prediction and error measures
- Powerful algorithmic framework that admits (blackbox) algorithms with certain properties (error-dependent comp. ratio, monotone)
- First results for weights, release dates, multiple machine models

Other scheduling results involving predictions

- load balancing [Lattanzi, Lavastida, Moseley, Vassilvitskii SODA 2020]
- flowtime minimization [Azar, Leonardi, Touitou STOC 2021, SODA 2022]
- speed scaling [Bamas, Maggiori, Rohwedder, Svensson NeurIPS 2020], [Antoniadis, Ganje, Shahkarami SWAT 2022]

Online Routing Problems

more general online graph problems incl. network design

Joint work with Giulia Bernardini (Trieste), Alexander Lindermayr (Bremen), Alberto Marchetti-Spaccamela (Rome), Leen Stougie & Michelle Sweering (CWI).

The Online Traveling Salesperson Problem

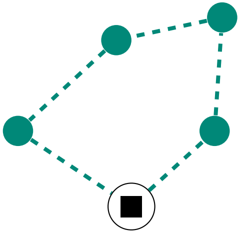
Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN

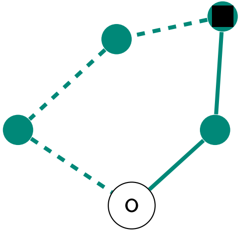


The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan

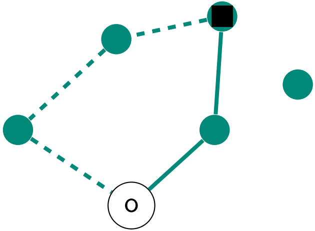


The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan

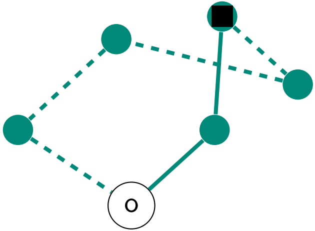


The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN

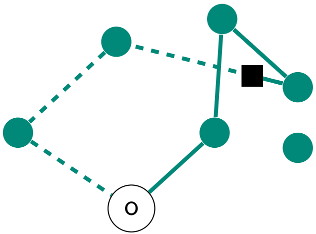


The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

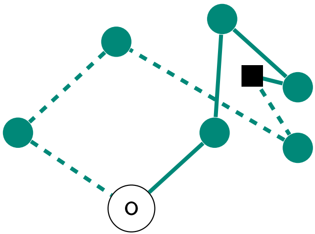
REPLAN



The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

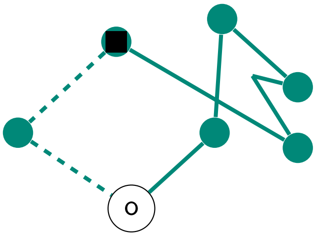
Replan



The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN

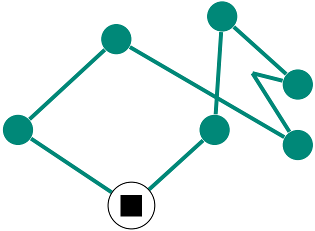


The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN

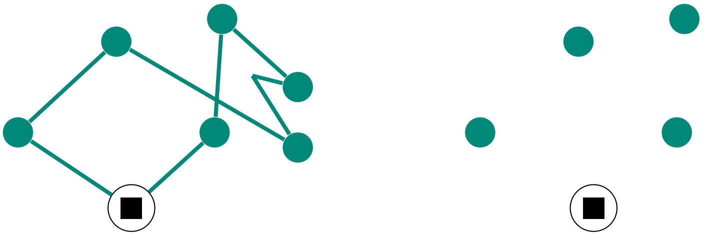


The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

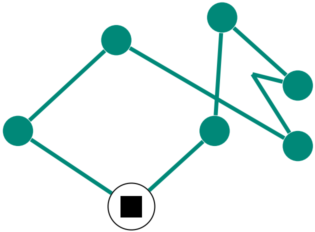
REPLAN IGNORE



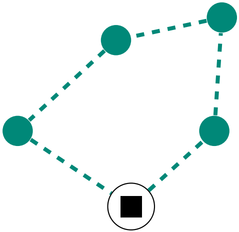
The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



Ignore

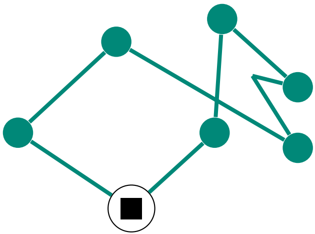


The Online Traveling Salesperson Problem

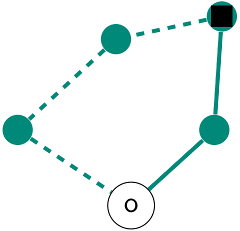
Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



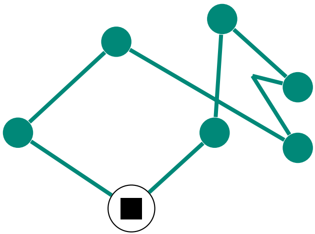
Ignore



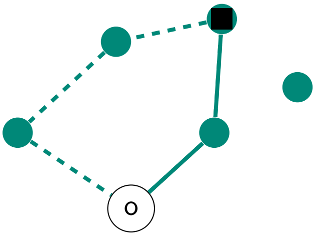
The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan)
visiting every request after its arrival and returning to the origin.

Replan



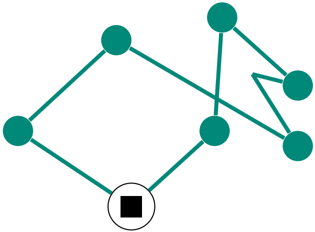
Ignore



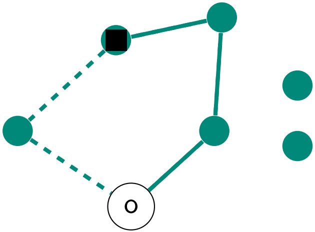
The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



Ignore

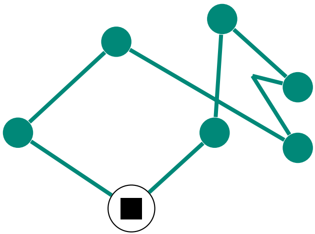


The Online Traveling Salesperson Problem

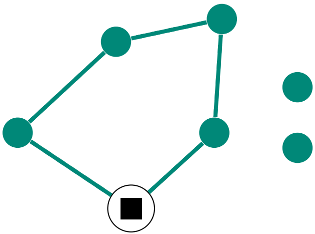
Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



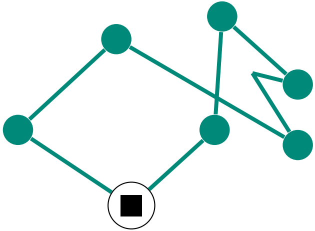
Ignore



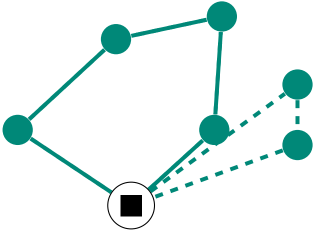
The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



Ignore

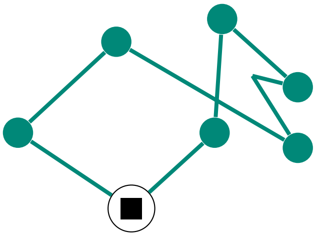


The Online Traveling Salesperson Problem

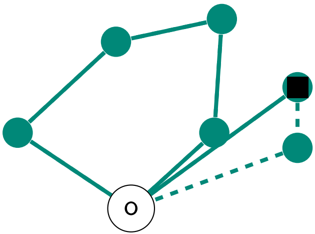
Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



Ignore

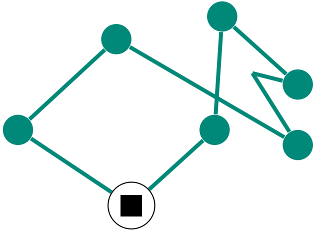


The Online Traveling Salesperson Problem

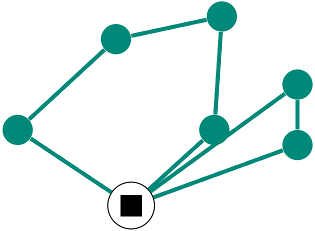
Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

Replan



Ignore

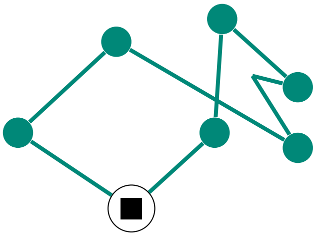


The Online Traveling Salesperson Problem

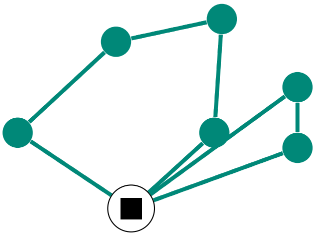
Input: Requests (x, r) arrive online at time r at location x

Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN



IGNORE

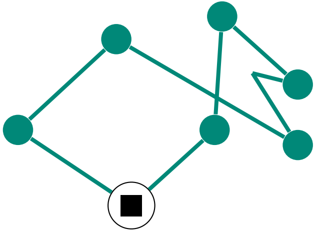


- REPLAN and IGNORE: 2.5-competitive [Ausiello et al. 2001]

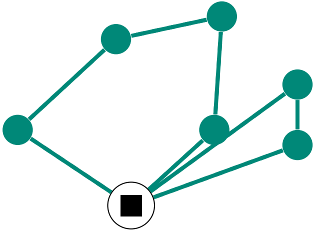
The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN



IGNORE

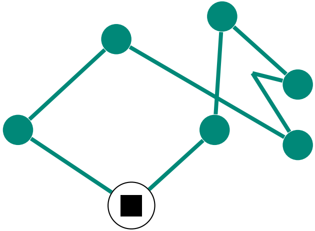


- REPLAN and IGNORE: 2.5-competitive [Ausiello et al. 2001]
- SMARTSTART: 2-competitive, best possible [Ascheuer et al. 2000]

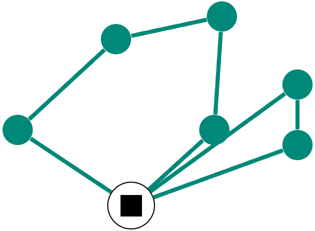
The Online Traveling Salesperson Problem

Input: Requests (x, r) arrive online at time r at location x
Task: Determine a tour of minimum total length (makespan) visiting every request after its arrival and returning to the origin.

REPLAN



IGNORE



- REPLAN and IGNORE: 2.5-competitive [Ausiello et al. 2001]
- SMARTSTART: 2-competitive, best possible [Ascheuer et al. 2000]

Generalization: Online Dial-a-Ride (tour for transportation requests)

Graph Problems with Predictions and Roadmap

Other works: graph problems with predictions

- Network design [Azar, Panigrahi, Touitou, SODA 2022], [Moseley, Xu, AAAI 2022]
- Graph exploration [Eberle, Lindermayr, Nölke, **M.**, Schlöter, AAAI 2022]
- Minimum Spanning Tree with queries [Erlebach, de Lima, **M.**, Schlöter, 2022]

Graph Problems with Predictions and Roadmap

Other works: graph problems with predictions

- Network design [Azar, Panigrahi, Touitou, SODA 2022], [Moseley, Xu, AAAI 2022]
- Graph exploration [Eberle, Lindermayr, Nölke, **M.**, Schlöter, AAAI 2022]
- Minimum Spanning Tree with queries [Erlebach, de Lima, **M.**, Schlöter, 2022]

Roadmap

1. Universal error measure for input predictions based on edge covers in suitably defined graphs
2. Algorithms with error-dependent competitive ratio
   - Online TSP (and Online Dial-a-Ride)
   - Online Steiner Tree (Online Facility Location, Steiner Forest)

Prediction Model

Prediction: request sequence (release dates and points/nodes)

Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

(i) location, (ii) time, and (iii) length of sequence

Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

(i) location, (ii) time, and (iii) length of sequence

Goals of an error measure:

- distinguish "good" and "bad" predictions
- quantify the effect of a erroneous prediction on the objective value of an algorithm trusting the predictions

Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

(i) location, (ii) time, and (iii) length of sequence

Goals of an error measure:

- distinguish "good" and "bad" predictions
- quantify the effect of a erroneous prediction on the objective value of an algorithm trusting the predictions

Intuition for detour error:

- Any "good" algorithm needs to trust predictions to some extent.

Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

(i) location, (ii) time, and (iii) length of sequence

Goals of an error measure:

- distinguish "good" and "bad" predictions
- quantify the effect of a erroneous prediction on the objective value of an algorithm trusting the predictions

Intuition for detour error:

- Any "good" algorithm needs to trust predictions to some extent.
- An unpredicted actual request arrives (unexpected).

Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

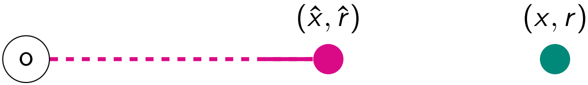
(i) location, (ii) time, and (iii) length of sequence

Goals of an error measure:

- distinguish "good" and "bad" predictions
- quantify the effect of a erroneous prediction on the objective value of an algorithm trusting the predictions

Intuition for detour error:

- Any "good" algorithm needs to trust predictions to some extent.
- An unpredicted actual request arrives (unexpected).



Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

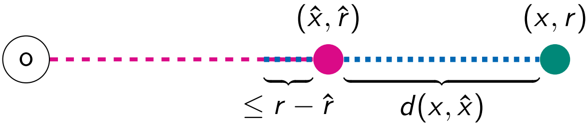
(i) location, (ii) time, and (iii) length of sequence

Goals of an error measure:

- distinguish "good" and "bad" predictions
- quantify the effect of a erroneous prediction on the objective value of an algorithm trusting the predictions

Intuition for detour error:

- Any "good" algorithm needs to trust predictions to some extent.
- An unpredicted actual request arrives (unexpected).



Prediction Model

Prediction: request sequence (release dates and points/nodes)

Possible errors in:

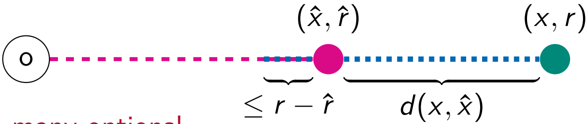
(i) location, (ii) time, and (iii) length of sequence

Goals of an error measure:

- distinguish "good" and "bad" predictions
- quantify the effect of a erroneous prediction on the objective value of an algorithm trusting the predictions

Intuition for detour error:

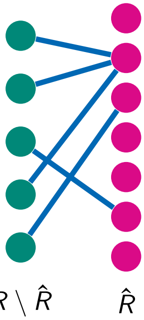
- Any "good" algorithm needs to trust predictions to some extent.
- An unpredicted actual request arrives (unexpected).



. . . might be many options!

Minimum-Cost Edge Cover

Idea: Model potential detours for serving unexpected requests as complete bipartite graph with edge cost $\gamma((x,r),(\hat{x},\hat{r})) = (r-\hat{r})_+ + d(x,\hat{x})$.

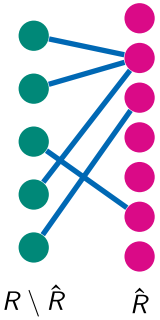


min-cost edge cover of
$R \setminus \hat{R}$ with value $\Gamma(R,\hat{R})$

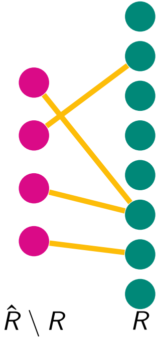
Minimum-Cost Edge Cover

Idea: Model potential detours for serving unexpected requests as complete bipartite graph with edge cost $\gamma((x,r),(\hat{x},\hat{r})) = (r-\hat{r})_+ + d(x,\hat{x})$.

Similarly, absent predicted requests can be covered by predicted requests.



min-cost edge cover of $R \setminus \hat{R}$ with value $\Gamma(R,\hat{R})$

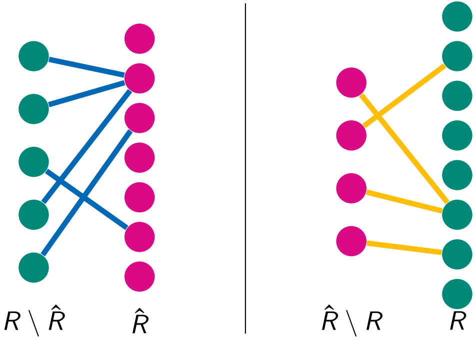


min-cost edge cover of $\hat{R} \setminus R$ with value $\Gamma(\hat{R},R)$

Minimum-Cost Edge Cover

Idea: Model potential detours for serving unexpected requests as complete bipartite graph with edge cost $\gamma((x,r),(\hat{x},\hat{r})) = (r-\hat{r})_+ + d(x,\hat{x})$.

Similarly, absent predicted requests can be covered by predicted requests.



min-cost edge cover of $R \setminus \hat{R}$ with value $\Gamma(R, \hat{R})$

min-cost edge cover of $\hat{R} \setminus R$ with value $\Gamma(\hat{R}, R)$

Edge Cover Error Measure: $\Lambda(\hat{R}, R) = \Gamma(R, \hat{R}) + \Gamma(\hat{R}, R)$

Learning-Augmented Algorithms

Common approach: combine online & offline algorithms in a clever way

$\rightarrow$ switching between algorithms might be expensive

Learning-Augmented Algorithms

Common approach: combine online & offline algorithms in a clever way
$\rightarrow$ switching between algorithms might be expensive

Algorithm DelayedTrust

$\hat{T}$: optimal tour on prediction; $\hat{C}$: its length; $\alpha > 0$ confidence param.

(i) Follow blackbox online algorithm $\mathcal{A}$ as long as for time t holds $t \leq \alpha \cdot \hat{C} - d(p(t), 0)$. (**robustness**)

(ii) Move the server to the origin.

(iii) Follow $\hat{T}$ and replan if actual unexpected request arrives. (**consist.**)

Learning-Augmented Algorithms

Common approach: combine online & offline algorithms in a clever way

$\rightarrow$ switching between algorithms might be expensive

Algorithm DelayedTrust

$\hat{T}$: optimal tour on prediction; $\hat{C}$: its length; $\alpha > 0$ confidence param.

(i) Follow blackbox online algorithm $\mathcal{A}$ as long as for time t holds $t \leq \alpha \cdot \hat{C} - d(p(t), 0)$. (**robustness**)

(ii) Move the server to the origin.

(iii) Follow $\hat{T}$ and replan if actual unexpected request arrives. (**consist.**)

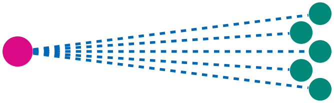
Theorem

For every $\alpha > 0$ and ρ-competitive online algorithm $\mathcal{A}$, DelayedTrust has a competitive ratio of at most

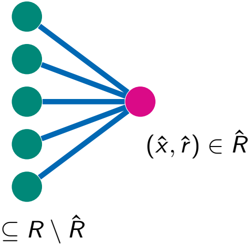
$$\min\left\{(1+\alpha)\left(1+\frac{2\cdot\Lambda}{\mathrm{OPT}}\right), 1+\left(1+\frac{1}{\alpha}\right)\cdot\rho\right\}.$$

Instance

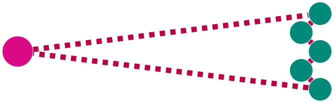
Instance



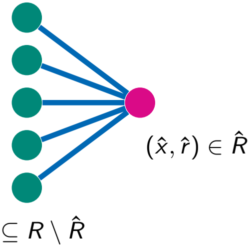
Edge cover bipartite graph



Instance

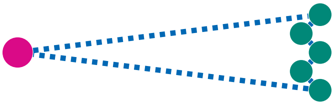


Edge cover bipartite graph

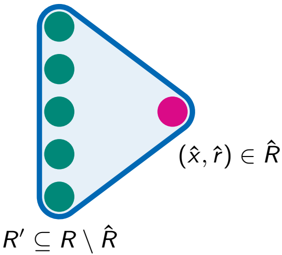


Refinement: Hyperedge Cover

Instance



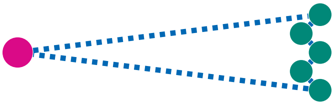
Edge cover bipartite graph



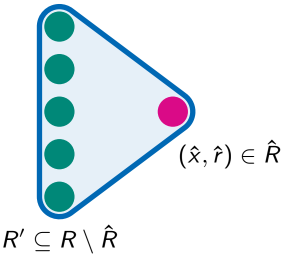
Hyperedge cost: length of opt. tour for R' which starts in $\hat{x}$ at time $\hat{r}$.

Refinement: Hyperedge Cover

Instance



Edge cover bipartite graph

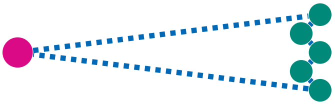


Hyperedge cost: length of opt. tour for R' which starts in $\hat{x}$ at time $\hat{r}$.

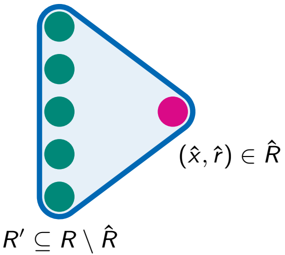
Hyperedge cover error: min-cost hyperedge cover using hyperedges of size at most k.

Refinement: Hyperedge Cover

Instance



Edge cover bipartite graph



Hyperedge cost: length of opt. tour for R' which starts in $\hat{x}$ at time $\hat{r}$.

Hyperedge cover error: min-cost hyperedge cover using hyperedges of size at most k.

$\rightarrow$ configurable, stronger, admits refined bounds for our algorithm

Further Applications

- Universal hyperedge cover error for online graph (metric) problems
  - captures error in # requests, location and time
  - nice properties, configurable, seems to capture “true” error very well

Further Applications

- Universal hyperedge cover error for online graph (metric) problems
  - captures error in # requests, location and time
  - nice properties, configurable, seems to capture “true” error very well
- Bounds for (new and old) learning-augmented algorithms
  - first framework for online TSP and Dial-a-Ride (online-time)
  - new bounds for known algorithm for network design (online-list)

[Azar, Panigrahi, Touitou SODA 2022]

Setting	Problem	Algorithm	Error-dependency
online-time	TSP	SmartTrust	$(1+\alpha)\cdot \mathrm{OPT} + 3 \cdot \Lambda_k$
	DARP	SmartTrust	$(1+\alpha)\cdot \mathrm{OPT} + 3 \cdot \Lambda_k$
online-list	Steiner Tree	APT	$\mathcal{O}(1)\cdot \mathrm{OPT} + \mathcal{O}(\log(k)) \cdot \Lambda_k$
	Steiner Forest	APT	$\mathcal{O}(1)\cdot \mathrm{OPT} + \mathcal{O}(k) \cdot \Lambda_k$
	Facility Location	APT	$\mathcal{O}(1)\cdot \mathrm{OPT} + \mathcal{O}(\log(k)) \cdot \Lambda_k$

Future & Discussion

This is a fascinating line of research with many open directions!

Future & Discussion

This is a fascinating line of research with many open directions!

Some further directions

- Universal error measure for input predictions – further applications?

Future & Discussion

This is a fascinating line of research with many open directions!

Some further directions

- Universal error measure for input predictions – further applications?
- Formalize what constitutes a good error measure

Future & Discussion

This is a fascinating line of research with many open directions!

Some further directions

- Universal error measure for input predictions – further applications?
- Formalize what constitutes a good error measure
- Prediction models: input vs. action, predict less

Future & Discussion

This is a fascinating line of research with many open directions!

Some further directions

- Universal error measure for input predictions – further applications?
- Formalize what constitutes a good error measure
- Prediction models: input vs. action, predict less
- What is learnable? Integrate learning into algorithm design!

Future & Discussion

This is a fascinating line of research with many open directions!

Some further directions

- Universal error measure for input predictions – further applications?
- Formalize what constitutes a good error measure
- Prediction models: input vs. action, predict less
- What is learnable? Integrate learning into algorithm design!

Thank you.