Algorithms with Predictions

Learning-Augmented Algorithms for Scheduling Problems

Nicole Megow

Faculty of Mathematics and Computer Science
University of Bremen

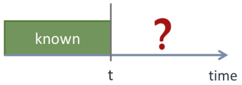
CO@Work 2024, Berlin

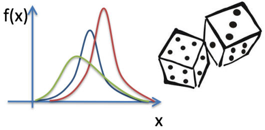
Optimziation under Uncertain Inputs

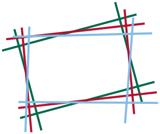
Optimziation under Uncertain Inputs

Different models for uncertain input

Online Information	Stochastic Information	Uncertainty sets

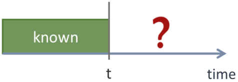
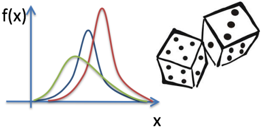
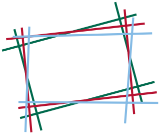






Optimziation under Uncertain Inputs

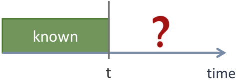
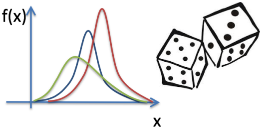
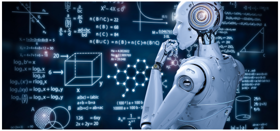
Different models for uncertain input

Online Information	Stochastic Information	Uncertainty sets
		

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

Optimziation under Uncertain Inputs

Different models for uncertain input

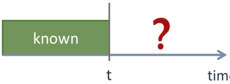
Online Information	Stochastic Information	Predictions (e.g. machine-learned)
		

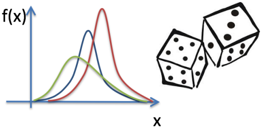
Different optimization frameworks: online optimization, stochastic optimization, robust optimization, etc.

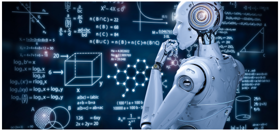
Optimziation under Uncertain Inputs

Different models for uncertain input

Online Information	Stochastic Information	Predictions (e.g. machine-learned)







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

Can imperfect predictions improve rigorous performance guarantees?

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]

n elements

8	11	14	16	18	25	30	36	40	43	46	49	50	53	54	56	59	60	63

$q = 16$

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]

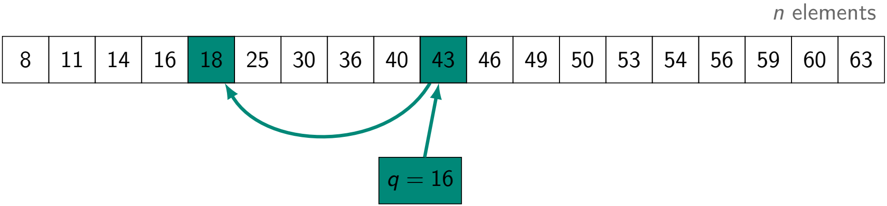
n elements

8	11	14	16	18	25	30	36	40	43	46	49	50	53	54	56	59	60	63

$q = 16$

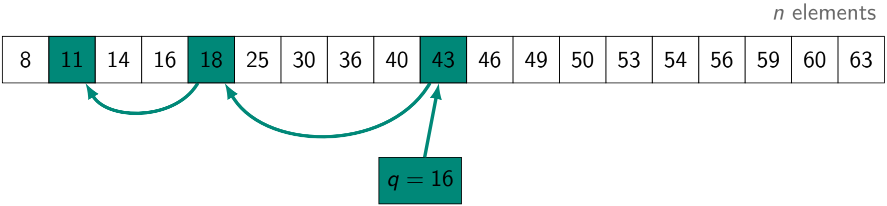
Motivating Example: Binary Search

[Kraska et al. SIGMOD 2018]

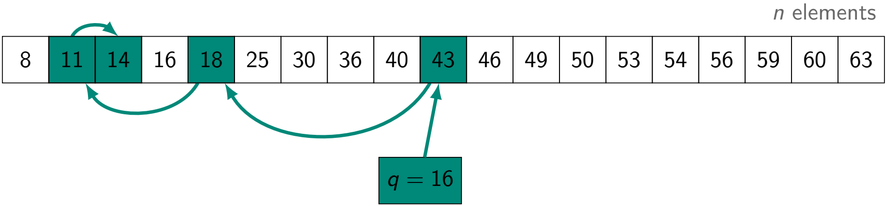


Motivating Example: Binary Search

[Kraska et al. SIGMOD 2018]

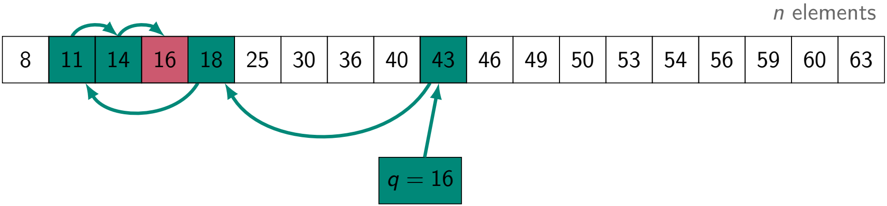


Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



Motivating Example: Binary Search

[Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]

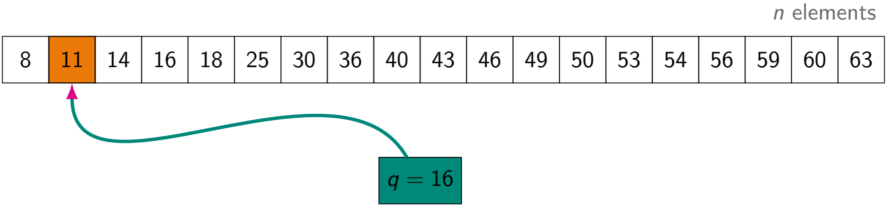
n elements

8	11	14	16	18	25	30	36	40	43	46	49	50	53	54	56	59	60	63

$q = 16$

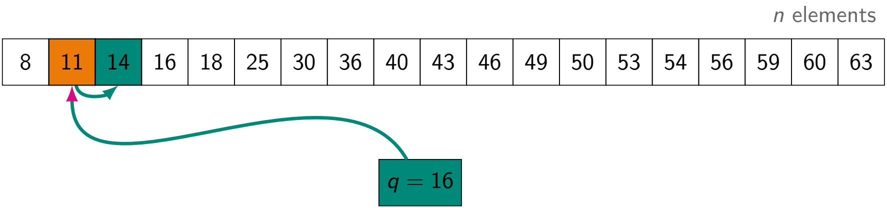
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



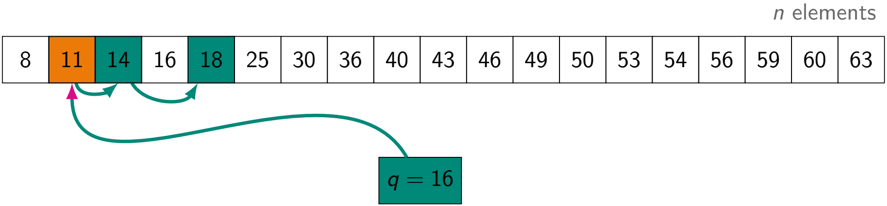
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - \mathit{index}(q)|$

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



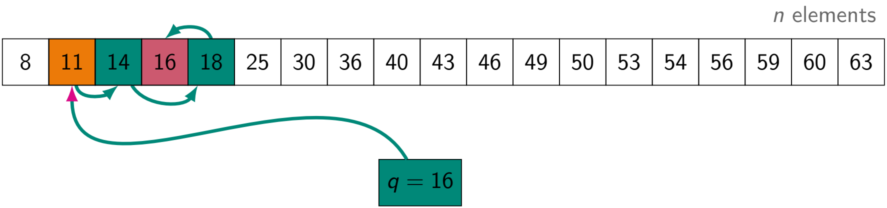
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



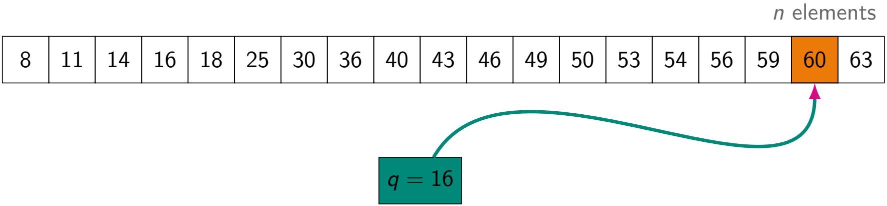
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



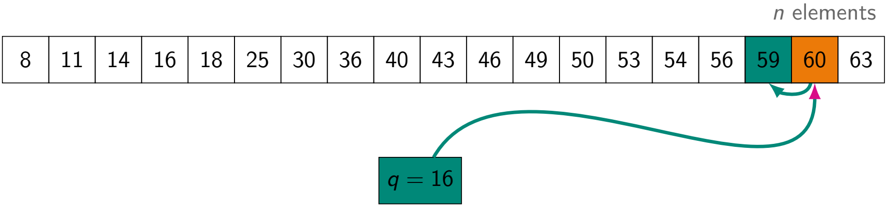
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



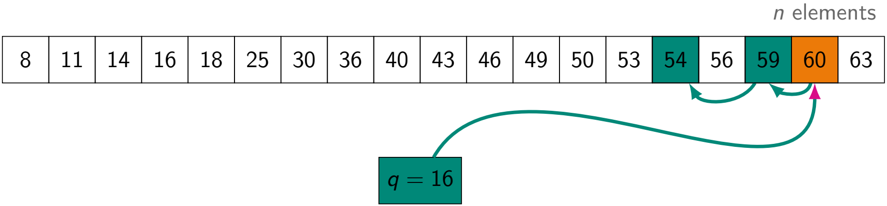
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



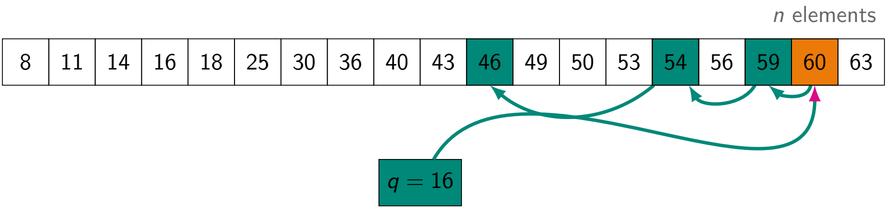
- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

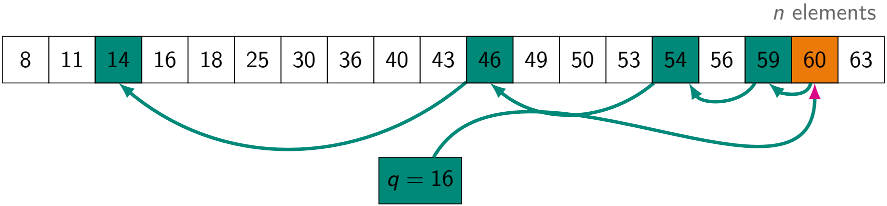
Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search

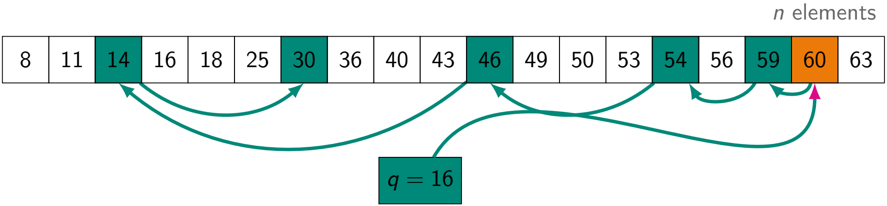
[Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

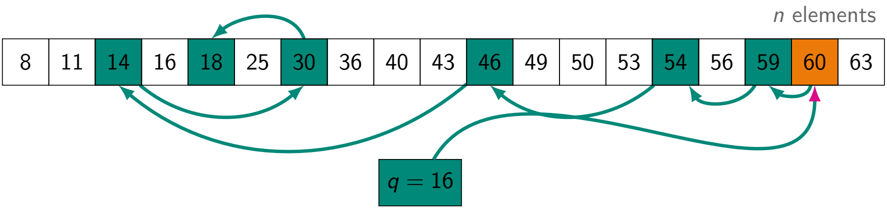
Motivating Example: Binary Search

[Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

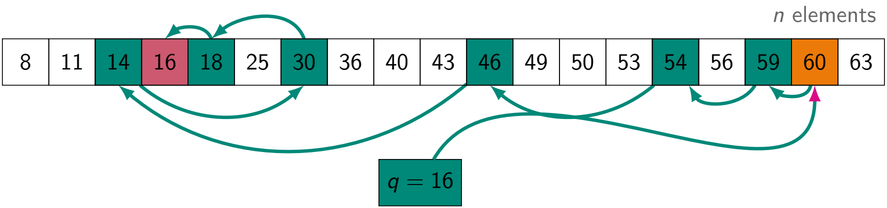
Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

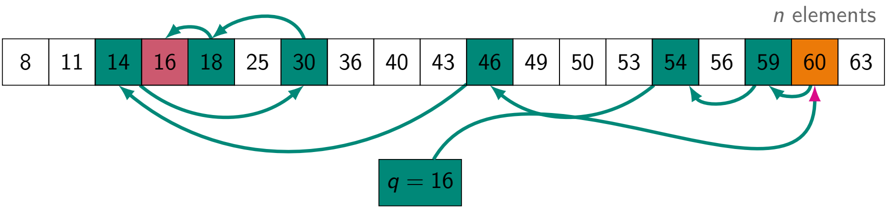
Motivating Example: Binary Search

[Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

Motivating Example: Binary Search [Kraska et al. SIGMOD 2018]



- Binary search: worst-case # queries is $\Theta(\log n)$
- Prediction: position $h(q)$ of target q
  - might be wrong; error $\eta = |h(q) - index(q)|$
  - from $h(q)$ use doubling to find other limit of search interval ($\Theta(\log \eta)$) and apply binary search within it ($\Theta(\log \eta)$)

$$\boxed{\Theta(\log n)} \xrightarrow{\text{predictions}} \boxed{\Theta(\log \eta)}$$

Learning-Augmented Algorithms

- Assume access to predictions (e.g. ML)
- Prediction is imperfect
- No information about their quality

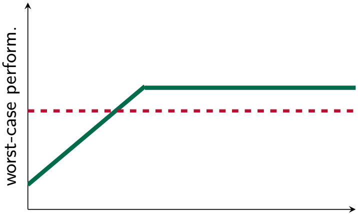
Learning-Augmented Algorithms

- Assume access to predictions (e.g. ML)
- Prediction is imperfect
- No information about their quality

© Adobe Stock

Desired algorithm properties

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



Learning-Augmented Algorithms

- Assume access to predictions (e.g. ML)
- Prediction is imperfect
- No information about their quality

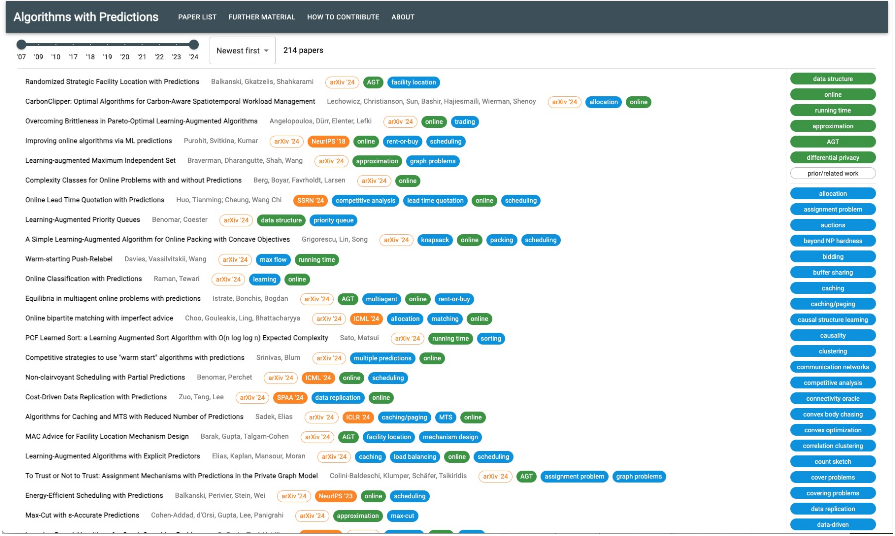
© Adobe Stock

Desired algorithm properties

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

Line of research (re)initiated by [Lykouris, Vassilvitskii (ICML 2018)], [Kraska et al. (SIGMOD 2018)] – became an extremely vibrant area

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



hosted by Alex Lindermayr and M.

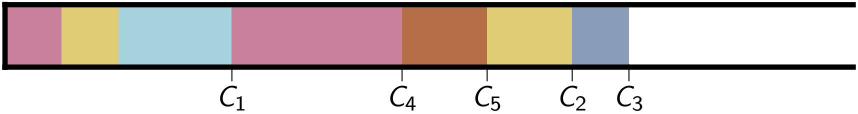
Min-Sum Scheduling

Input: set of jobs with processing requirements p_j

Min-Sum Scheduling

Input: set of jobs with processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine

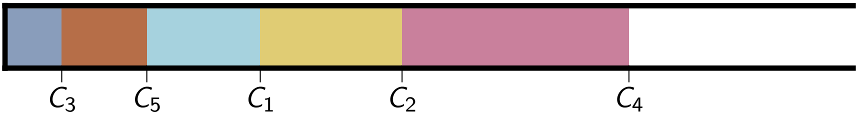


Objective: Minimize sum of completion times $\sum_j C_j$

Min-Sum Scheduling

Input: set of jobs with processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine



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

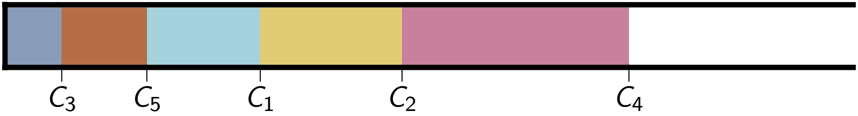
Optimal Schedule: [Smith 1956]

Shortest Processing Time first (SPT)

Min-Sum Scheduling

Input: set of jobs with processing requirements p_j

Goal: schedule jobs (preemptively) on a single machine



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

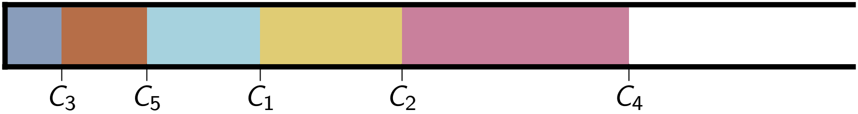
Optimal Schedule: [Smith 1956]

Shortest Processing Time first (SPT) $\frac{w_1}{p_1} \geq \ldots \geq \frac{w_n}{p_n}$ (WSPT)

Min-Sum Scheduling

Input: set of jobs with 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: [Smith 1956]

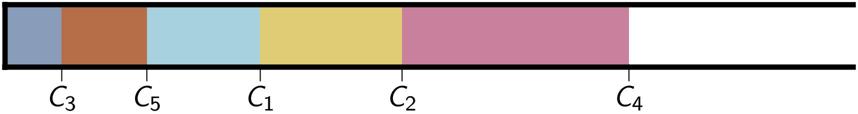
Shortest Processing Time first (SPT) $\frac{w_1}{p_1} \geq \ldots \geq \frac{w_n}{p_n}$ (WSPT)

Unrelated machines $(R|r_j, pmtn|\sum w_j C_j)$: 1.99-approximation [Im, Li 2016]

Min-Sum Scheduling

Input: set of jobs with 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: [Smith 1956]

Shortest Processing Time first (SPT) $\frac{w_1}{p_1} \geq \ldots \geq \frac{w_n}{p_n}$ (WSPT)

Unrelated machines $(R|r_j, pmtn|\sum w_j C_j)$: 1.99-approximation [Im, Li 2016]

Suppose processing times are unknown! (non-clairvoyant scheduling)

Non-Clairvoyant Min-Sum Scheduling

Processing times are unknown.

We cannot expect to find the optimal solution.

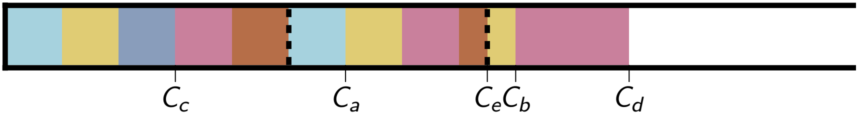
Competitive analysis (worst-case analysis)

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:

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

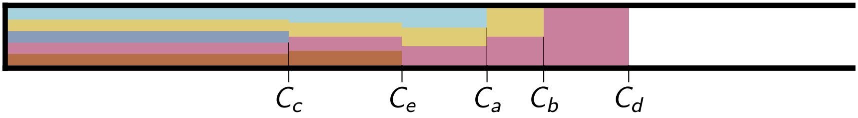
Non-Clairvoyant Min-Sum Scheduling

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



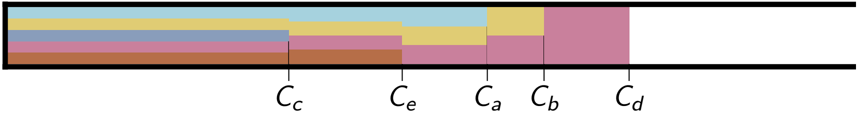
Non-Clairvoyant Min-Sum Scheduling

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



Non-Clairvoyant Min-Sum Scheduling

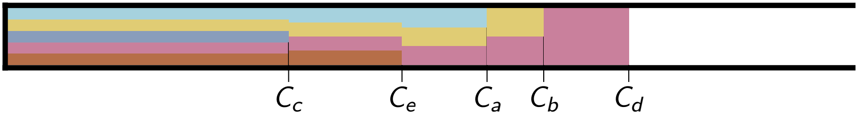
Round-Robin (RR) is 2-competitive for minimizing $\sum_j C_j$ on a single machine, and this is best-possible. [Motwani, Phillips, Torng 1994]



Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

Non-Clairvoyant Min-Sum Scheduling

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

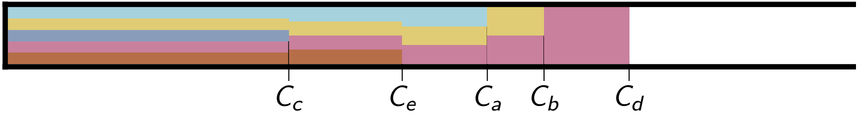


Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

$$C_1 = n \cdot p_1$$

Non-Clairvoyant Min-Sum Scheduling

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



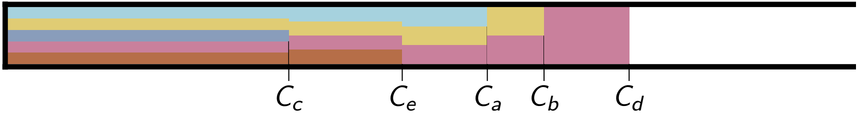
Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

$$C_1 = n \cdot p_1$$

$$C_2 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) = p_1 + (n-1) \cdot p_2$$

Non-Clairvoyant Min-Sum Scheduling

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



Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

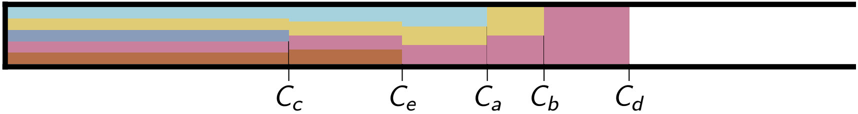
$$C_1 = n \cdot p_1$$

$$C_2 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) = p_1 + (n-1) \cdot p_2$$

$$C_3 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) + (n-2) \cdot (p_3 - p_2 - p_1) \leq p_1 + p_2 + (n-2) \cdot p_3$$

Non-Clairvoyant Min-Sum Scheduling

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



Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

$$C_1 = n \cdot p_1$$

$$C_2 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) = p_1 + (n-1) \cdot p_2$$

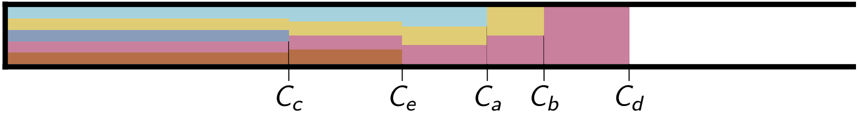
$$C_3 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) + (n-2) \cdot (p_3 - p_2 - p_1) \leq p_1 + p_2 + (n-2) \cdot p_3$$

$$\ldots$$

$$\sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} \left(\sum_{\ell=1}^{j-1} p_\ell + (n-j+1) \cdot p_j \right)$$

Non-Clairvoyant Min-Sum Scheduling

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



Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

$$C_1 = n \cdot p_1$$

$$C_2 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) = p_1 + (n-1) \cdot p_2$$

$$C_3 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) + (n-2) \cdot (p_3 - p_2 - p_1) \leq p_1 + p_2 + (n-2) \cdot p_3$$

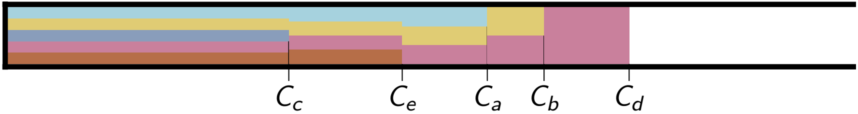
$$\ldots$$

$$\sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} \left(\sum_{\ell=1}^{j-1} p_\ell + (n-j+1) \cdot p_j \right)$$

$$\leq 2 \cdot \sum_{j=1}^{n} (n-j+1) \cdot p_j$$

Non-Clairvoyant Min-Sum Scheduling

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



Proof. Let $p_1 \leq p_2 \leq \ldots \leq p_n$.

$$C_1 = n \cdot p_1$$

$$C_2 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) = p_1 + (n-1) \cdot p_2$$

$$C_3 = n \cdot p_1 + (n-1) \cdot (p_2 - p_1) + (n-2) \cdot (p_3 - p_2 - p_1) \leq p_1 + p_2 + (n-2) \cdot p_3$$

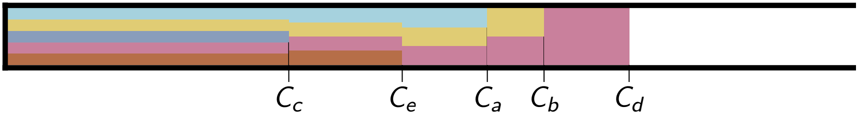
$\ldots$

$$\sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} \left(\sum_{\ell=1}^{j-1} p_\ell + (n-j+1) \cdot p_j \right)$$

$$\leq 2 \cdot \sum_{j=1}^{n} (n-j+1) \cdot p_j = 2 \cdot \mathrm{SPT} \qquad \square$$

Non-Clairvoyant Min-Sum Scheduling

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



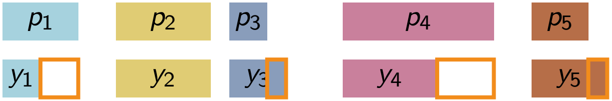
Further **Time-Sharing** algorithms for more general problems:

- 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-comp., 4.62-comp.) [Im, Kulkarni, Munagala 2018], [Lindermayr, **M.**, Jäger 2024]

Prediction Model

Prediction Model

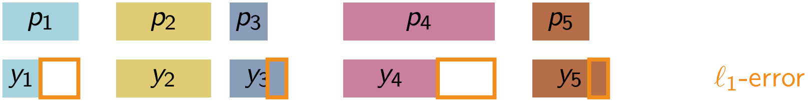
Predict job lengths y_j [Kumar, Purohit, Svitkina, NIPS 2018]



ℓ_1-error

Prediction Model

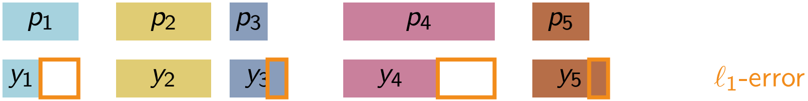
Predict job lengths y_j [Kumar, Purohit, Svitkina, NIPS 2018]



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

Prediction Model

Predict job lengths y_j [Kumar, Purohit, Svitkina, NIPS 2018]

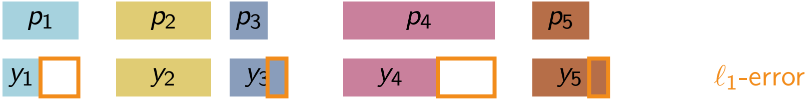


Error: $\ell_1 = \sum_{i=1}^{n} |p_j - y_j|$

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

Prediction Model

Predict job lengths y_j [Kumar, Purohit, Svitkina, NIPS 2018]



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

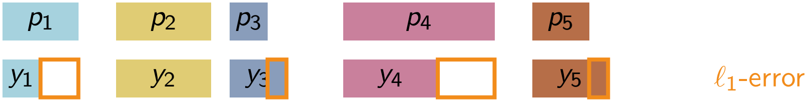
Natural algorithm: run shortest predicted job first (SPF).
("Follow the prediction")

Lemma

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

Prediction Model

Predict job lengths y_j [Kumar, Purohit, Svitkina, NIPS 2018]



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

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

Lemma

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).

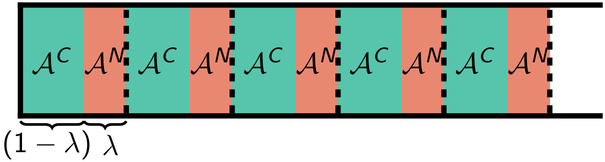
Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018], [Lindermayr, **M.** 2022]

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 ($\lambda, \mathcal{A}^C, \mathcal{A}^N$): run both $\mathcal{A}^C$ and $\mathcal{A}^N$



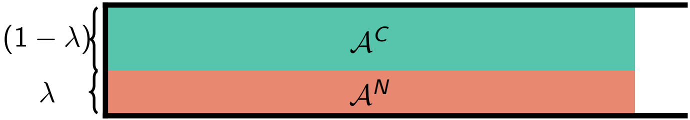
Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018], [Lindermayr, **M.** 2022]

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 $(\lambda, \mathcal{A}^C, \mathcal{A}^N)$: run both $\mathcal{A}^C$ and $\mathcal{A}^N$



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

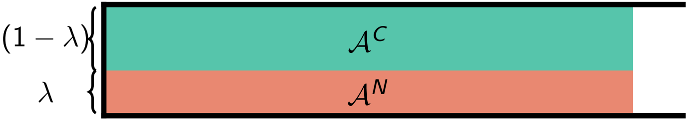
Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018], [Lindermayr, **M.** 2022]

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 $(\lambda, \mathcal{A}^C, \mathcal{A}^N)$: run both $\mathcal{A}^C$ and $\mathcal{A}^N$



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

Analysis: slowed down execution by factors $\frac{1}{1-\lambda}$ resp. $\frac{1}{\lambda}$

Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018] [Lindermayr, **M.** 2022]

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.

Preferential Time Sharing Framework (PTS)

[Kumar, Purohit, Svitkina 2018] [Lindermayr, **M.** 2022]

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 with 2-competitive RR and SPF achieves a competitive ratio of min$\{ \frac{1}{1-\lambda}(1 + \frac{n \cdot \ell_1}{\mathrm{OPT}}), \frac{2}{\lambda} \}$, for $\lambda \in (0, 1)$.

Preferential Time Sharing Framework (PTS)

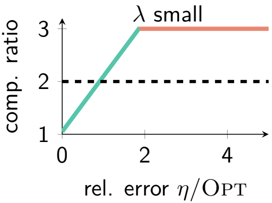
[Kumar, Purohit, Svitkina 2018] [Lindermayr, **M.** 2022]

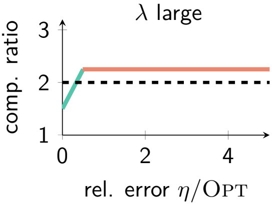
Theorem

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 with 2-competitive RR and SPF achieves a competitive ratio of $\min\{\frac{1}{1-\lambda}(1 + \frac{n \cdot \ell_1}{\mathrm{OPT}}), \frac{2}{\lambda}\}$, for $\lambda \in (0,1)$.





Powerful Framework?!

Yes, works for more general scheduling problems [Lindermayr, **M.** 2022]

Roadmap

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

Powerful Framework?!

Yes, works for more general scheduling problems [Lindermayr, **M.** 2022]

Roadmap

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

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

Powerful Framework?!

Yes, works for more general scheduling problems [Lindermayr, **M.** 2022]

Roadmap

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

- Proving error-dependent bounds seems difficult with ℓ_1-error (linear error vs. quadratic objective)
  $\rightarrow$ alternative error measures [Im et al. 2021], [Lindermayr, **M.** 2022]

Powerful Framework?!

Yes, works for more general scheduling problems [Lindermayr, **M.** 2022]

Roadmap

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

- Proving error-dependent bounds seems difficult with ℓ_1-error (linear error vs. quadratic objective)
  $\rightarrow$ alternative error measures [Im et al. 2021], [Lindermayr, **M.** 2022]
- Do we really need to predict all the job lengths?

Permutation Predictions

Lindermayr and **M.** 2022

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

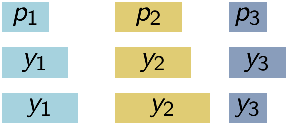
Motivation: knowing WSPT order is often sufficient for good approximations:

- 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

Lindermayr and **M.** 2022

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



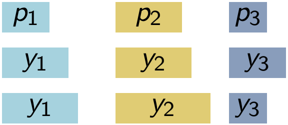
(W)SPT order $p_3 \leq p_1 \leq p_2$

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

Permutation Predictions

Lindermayr and **M.** 2022

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 $\ell_1, \nu > 0$

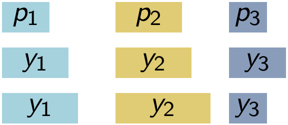
Error measure: quantifies effect of inversions $\mathcal{I}$ between $\hat{\sigma}$ and true WSPT order on list scheduling according to predicted order:

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

Permutation Predictions

Lindermayr and **M.** 2022

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:

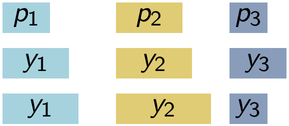
$$\eta^S = \sum_{(i,j) \in \mathcal{I}} (w_i p_j - w_j p_i)$$

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

Permutation Predictions

Lindermayr and **M.** 2022

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.

Scheduling on a Single Machine

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

1. prediction-clairvoyant $\mathcal{A}^C$: WSPT (optimal) [Smith56]

Schedule jobs in WSPT order

Scheduling on a Single Machine

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

1. prediction-clairvoyant $\mathcal{A}^C$: WSPT (optimal) [Smith56]

Schedule jobs in **predicted** order

Scheduling on a Single Machine

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

1. prediction-clairvoyant $\mathcal{A}^C$: WSPT (optimal) [Smith56]

   Schedule jobs in **predicted** order

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

Scheduling on a Single Machine

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

1. prediction-clairvoyant $\mathcal{A}^C$: WSPT (optimal) [Smith56]

   Schedule jobs in **predicted** order

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

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

Scheduling on a Single Machine

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

1. prediction-clairvoyant $\mathcal{A}^C$: WSPT (optimal) [Smith56]

   Schedule jobs in **predicted** order

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

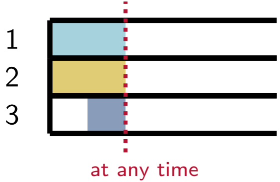
$$\min \left\{ \frac{1}{1-\lambda} \left(1 + \frac{\eta^S}{\textsc{Opt}} \right), \frac{2}{\lambda} \right\}$$

WSPT order also useful for more general scheduling settings.

Scheduling on Identical Machines

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

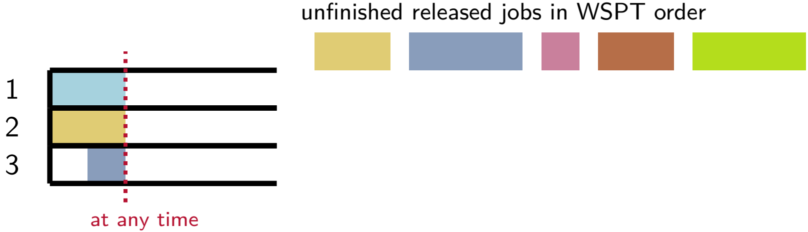
1. prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (2-competitive) [**M.**, Schulz 2004]



Scheduling on Identical Machines

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

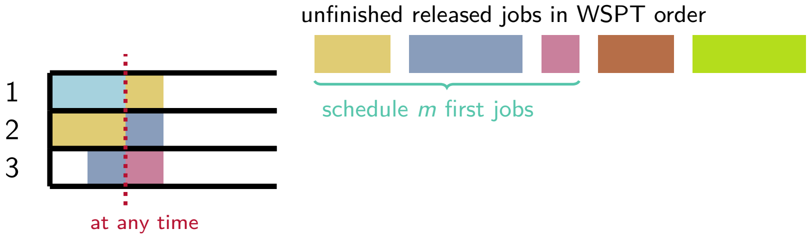
1. prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (2-competitive) [**M.**, Schulz 2004]



Scheduling on Identical Machines

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

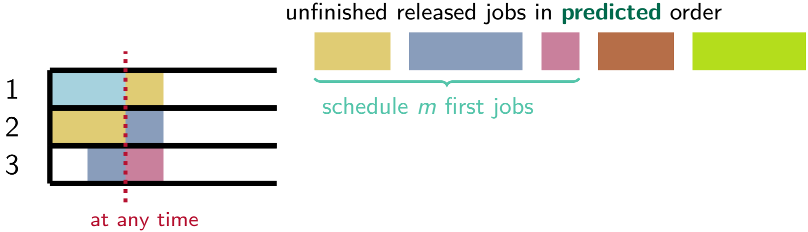
1. prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (2-competitive) [**M.**, Schulz 2004]



Scheduling on Identical Machines

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

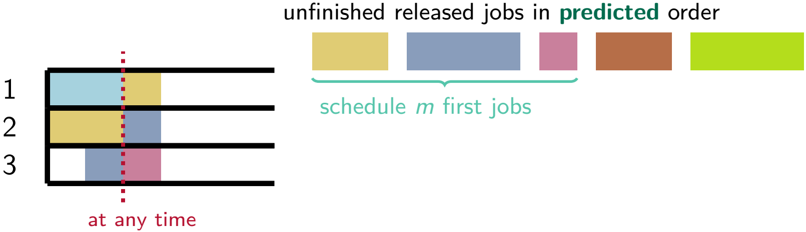
1. prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (2-competitive) [**M.**, Schulz 2004]



Scheduling on Identical Machines

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

1. prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (2-competitive) [**M.**, Schulz 2004]

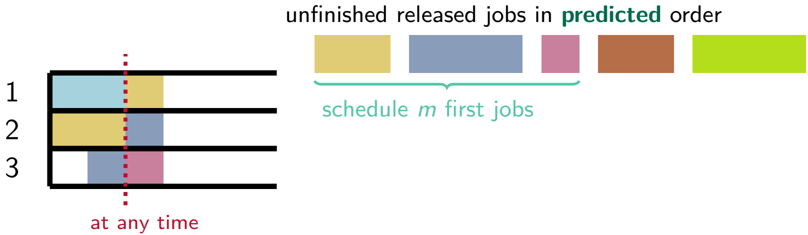


2. non-clairvoyant $\mathcal{A}^N$: WDEQ (3-competitive) [Beaumont et al. 2012]

Scheduling on Identical Machines

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

1. prediction-clairvoyant $\mathcal{A}^C$: P-WSPT (2-competitive) [**M.**, Schulz 2004]



2. non-clairvoyant $\mathcal{A}^N$: WDEQ (3-competitive) [Beaumont et al. 2012]

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

Scheduling on Unrelated Machines

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

Scheduling on Unrelated Machines

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

1. prediction-clairvoyant $\mathcal{A}^C$:

   Known algorithms may not always work!

   $\rightarrow$ prove error-dependent competitive ratio for alg. that trusts the prediction

Scheduling on Unrelated Machines

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

1. prediction-clairvoyant $\mathcal{A}^C$:

   Known algorithms may not always work!

   $\rightarrow$ prove error-dependent competitive ratio for alg. that trusts the prediction

   We can design such algorithm, but it is worse than the non-clairvoyant algorithm Proportional Fairness (PF).

Scheduling on Unrelated Machines

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

1. prediction-clairvoyant $\mathcal{A}^C$:

   Known algorithms may not always work!

   $\rightarrow$ prove error-dependent competitive ratio for alg. that trusts the prediction

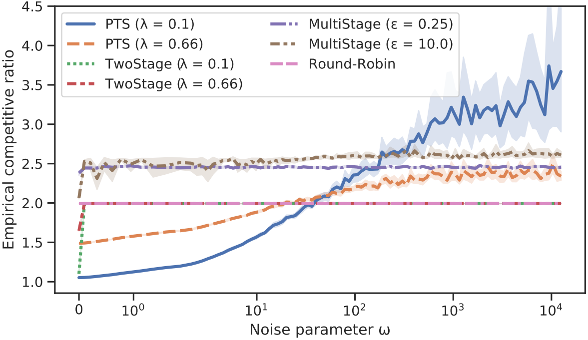
   We can design such algorithm, but it is worse than the non-clairvoyant algorithm Proportional Fairness (PF).

2. non-clairvoyant $\mathcal{A}^N$: Proportional Fairness (4.62-competitive)

[Lindermayr, **M.**, Jäger 2024]

Sensitivity Experiments

- Single machine, unweighted jobs
- Synthetic instances sampled from Pareto-distribution with shape 1.1
  Many small jobs and few very large jobs! (common)



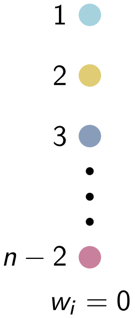
Online Precedence Constraints (min $\sum_j w_j C_j$)

Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors

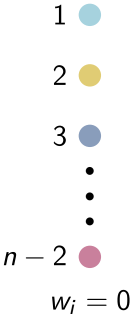
Online Precedence Constraints ($\min \sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors



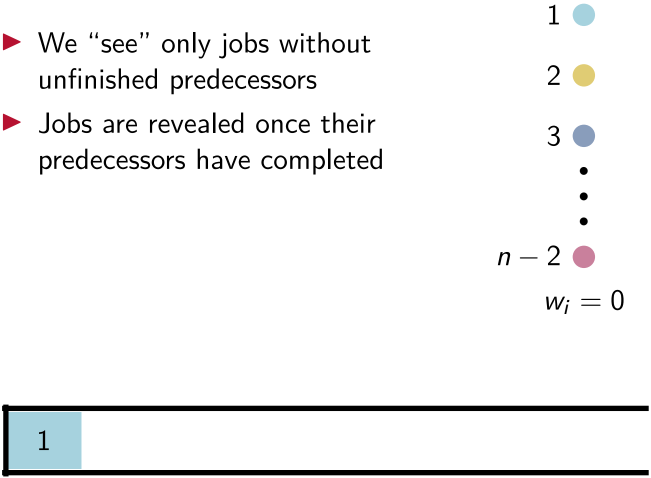
Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



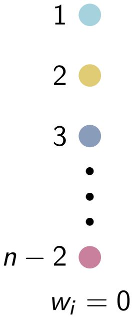
Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



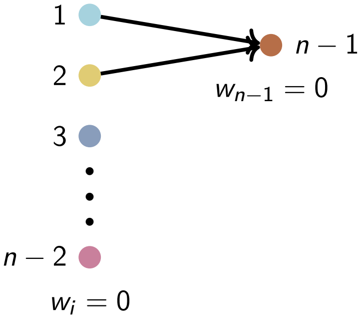
Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



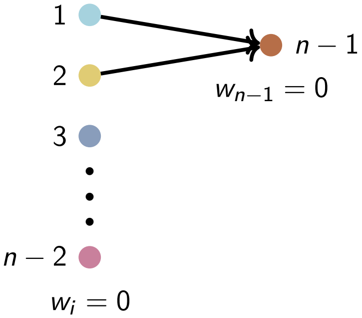
Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



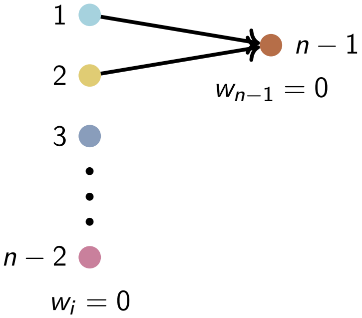
Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



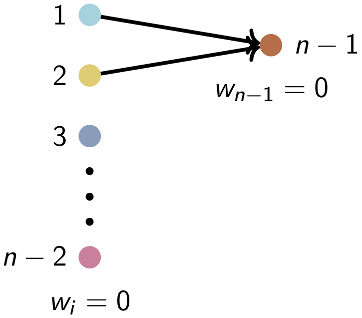
Online Precedence Constraints (min $\sum_j w_j C_j$)

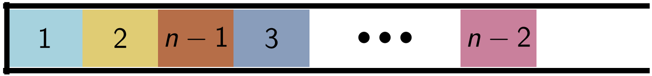
- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



Online Precedence Constraints (min $\sum_j w_j C_j$)

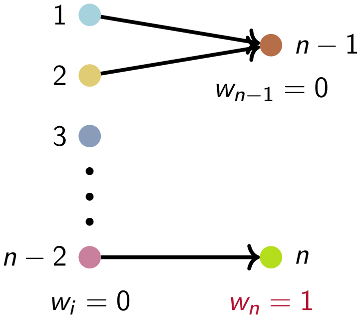
- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed

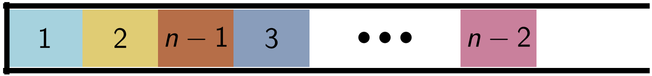




Online Precedence Constraints (min $\sum_j w_j C_j$)

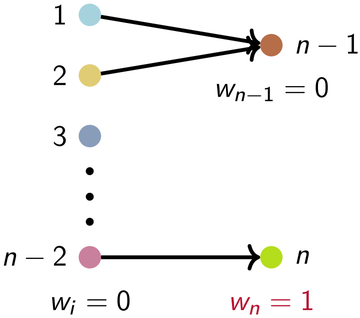
- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed

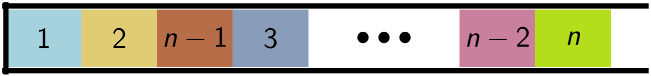




Online Precedence Constraints (min $\sum_j w_j C_j$)

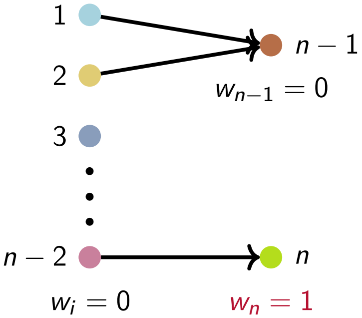
- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed

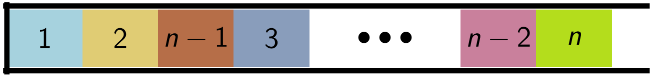




Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed

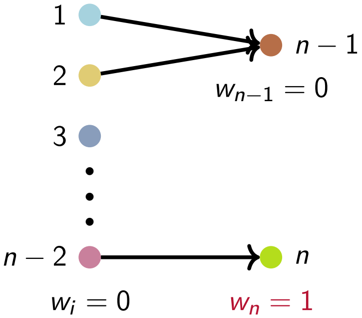


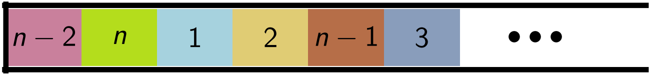


$\sum_{j=1}^{n} w_j C_j = n$

Online Precedence Constraints (min $\sum_j w_j C_j$)

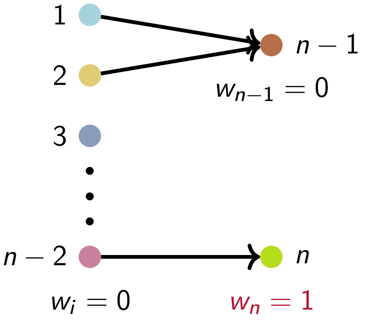
- We “see” only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed

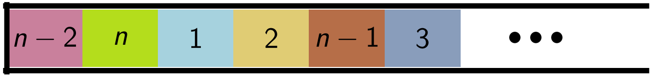




Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed



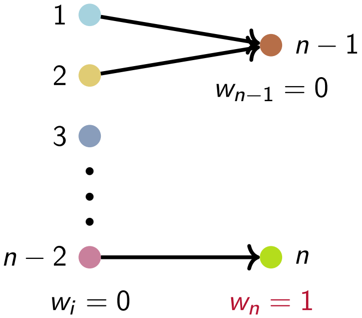


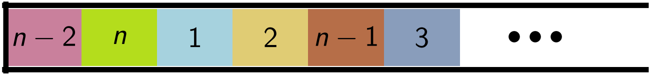
$\sum_{j=1}^{n} w_j C_j = 2$

Online Precedence Constraints (min $\sum_j w_j C_j$)

- We "see" only jobs without unfinished predecessors
- Jobs are revealed once their predecessors have completed

Any online algorithm has a competitive ratio $\Omega(n)$!





$\sum_{j=1}^{n} w_j C_j = 2$

Full Input Prediction – PTS Framework

[Lassota, Lindermayr, **M.**, Schlöter, ICML 2023]

Predict the full instance or a permutation of jobs

Full Input Prediction – PTS Framework

[Lassota, Lindermayr, **M.**, Schlöter, ICML 2023]

Predict the full instance or a permutation of jobs

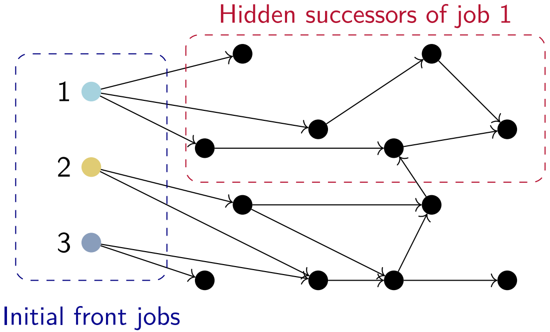
1. "Follow-the-Prediction" is $(1+\eta)$-competitive (η permutation error)

Full Input Prediction – PTS Framework

[Lassota, Lindermayr, **M.**, Schlöter, ICML 2023]

Predict the full instance or a permutation of jobs

1. "Follow-the-Prediction" is $(1+\eta)$-competitive (η permutation error)
2. Robustness via Round Robin for front jobs: ω-competitive (ω width)

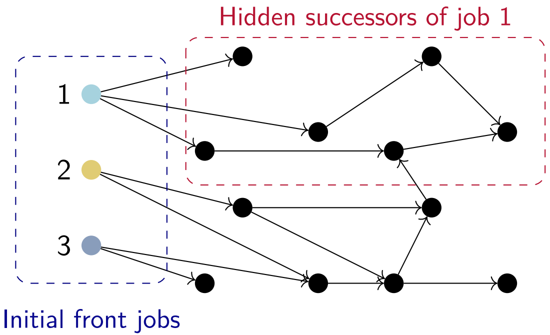


Full Input Prediction – PTS Framework

[Lassota, Lindermayr, **M.**, Schlöter, ICML 2023]

Predict the full instance or a permutation of jobs

1. "Follow-the-Prediction" is $(1+\eta)$-competitive (η permutation error)
2. Robustness via Round Robin for front jobs: ω-competitive (ω width)

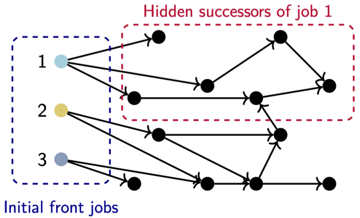


Theorem. Preferential Time Sharing is $\mathcal{O}(\min\{1+\eta, \omega\})$-competitive.

Minimalistic Input Prediction

What additional information is needed to improve upon lower bound?

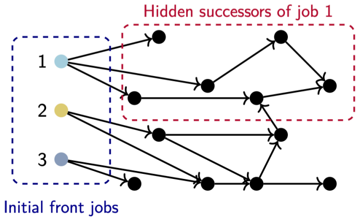
$\rightarrow$ exact information!



Minimalistic Input Prediction

What additional information is needed to improve upon lower bound?

→ exact information!

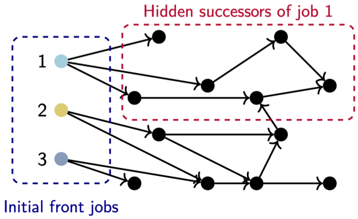


Full input

Minimalistic Input Prediction

What additional information is needed to improve upon lower bound?

→ exact information!



Full input

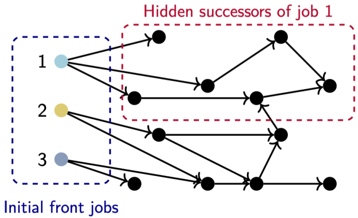
Total successor weight

Avg. successor weight

Minimalistic Input Prediction

What additional information is needed to improve upon lower bound?

→ exact information!



Full input

Total successor weight

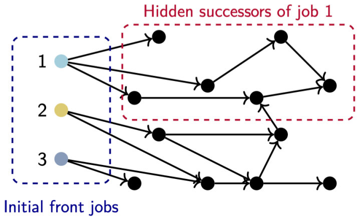
Avg. successor weight

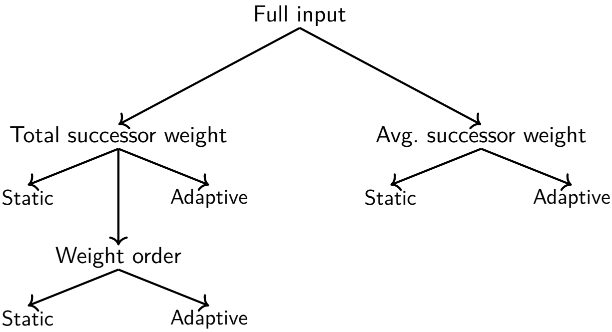
Weight order

Minimalistic Input Prediction

What additional information is needed to improve upon lower bound?

→ exact information!

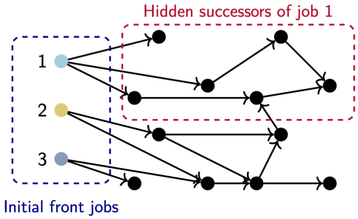


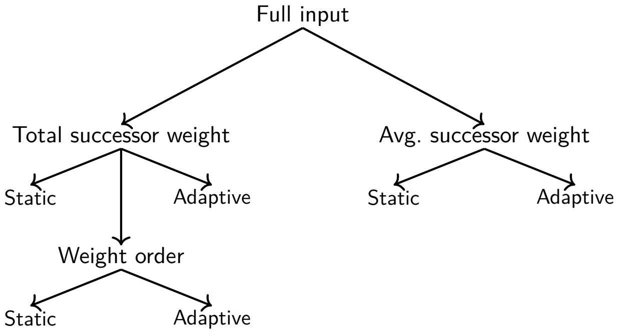


Minimalistic Input Prediction

What additional information is needed to improve upon lower bound?

→ exact information!

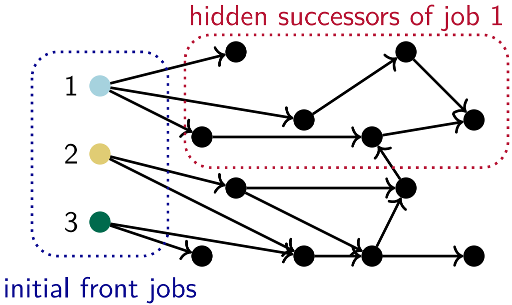




… results for different topologies – not today.

Weight Prediction based on Graph Decomposition

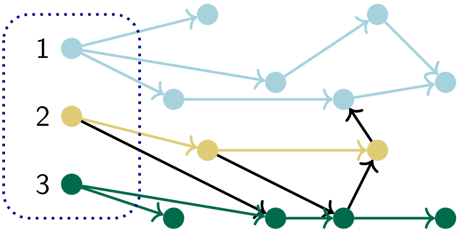
[Jäger & Warode, 2024]



Weight Prediction based on Graph Decomposition

[Jäger & Warode, 2024]

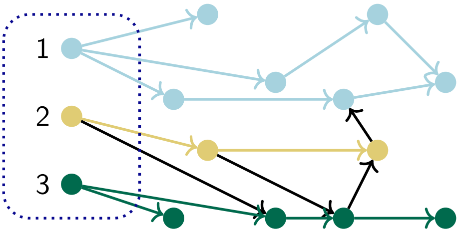
- Decompose DAG G into out-trees rooted at the front jobs



Weight Prediction based on Graph Decomposition

[Jäger & Warode, 2024]

- Decompose DAG G into out-trees rooted at the front jobs
- Algorithm: run front jobs v at a rate proportional to the total weight $w(T(v))$ of jobs in v's tree $T(v)$

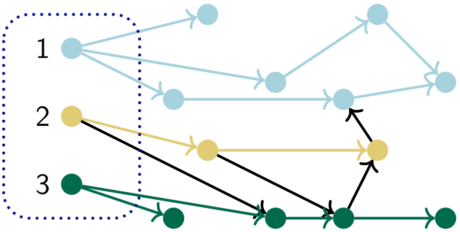


Weight Prediction based on Graph Decomposition

[Jäger & Warode, 2024]

- Decompose DAG G into out-trees rooted at the front jobs
- Algorithm: run front jobs v at a rate proportional to the total weight $w(T(v))$ of jobs in v's tree $T(v)$

Theorem (Jäger & Warode 2024). This algorithm is 2-competitive.

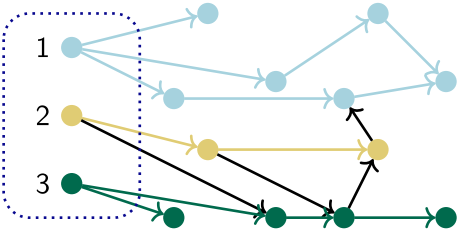


Weight Prediction based on Graph Decomposition

[Jäger & Warode, 2024]

- Decompose DAG G into out-trees rooted at the front jobs
- Algorithm: run front jobs v at a rate proportional to the total weight $w(T(v))$ of jobs in v's tree $T(v)$

Theorem (Jäger & Warode 2024). This algorithm is 2-competitive.



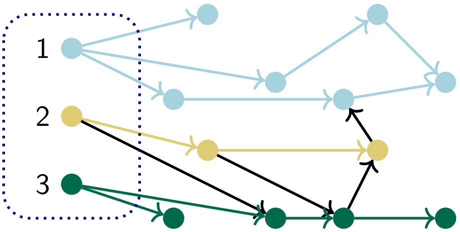
- Prediction: the total weight $w(T(v))$ of successors of job v in T

Weight Prediction based on Graph Decomposition

[Jäger & Warode, 2024]

- Decompose DAG G into out-trees rooted at the front jobs
- Algorithm: run front jobs v at a rate proportional to the total weight $w(T(v))$ of jobs in v's tree $T(v)$

Theorem (Jäger & Warode 2024). This algorithm is 2-competitive.



- Prediction: the total weight $w(T(v))$ of successors of job v in T

Theorem. The time sharing framework is $\mathcal{O}(\min\{2+\eta, \omega\}$-competitive.

Summary and Outlook

This talk

- Algorithms with predictions
  binary search, scheduling with unknown job sizes, precedences (online)

Summary and Outlook

This talk

- Algorithms with predictions
  binary search, scheduling with unknown job sizes, precedences (online)

- Powerful time-sharing framework
  admits (blackbox) algorithms with error-dependent performance guarantee

Summary and Outlook

This talk

- Algorithms with predictions
  binary search, scheduling with unknown job sizes, precedences (online)
- Powerful time-sharing framework
  admits (blackbox) algorithms with error-dependent performance guarantee
- Prediction models and error measures
  predictions: length, permutation, weight decomp.; errors: ℓ_1 error, and more

Summary and Outlook

This talk

- Algorithms with predictions
  binary search, scheduling with unknown job sizes, precedences (online)
- Powerful time-sharing framework
  admits (blackbox) algorithms with error-dependent performance guarantee
- Prediction models and error measures
  predictions: length, permutation, weight decomp.; errors: ℓ_1 error, and more

Outlook:

- More sophisticated techniqies to leverage imperfect predictions

Summary and Outlook

This talk

- Algorithms with predictions
  binary search, scheduling with unknown job sizes, precedences (online)
- Powerful time-sharing framework
  admits (blackbox) algorithms with error-dependent performance guarantee
- Prediction models and error measures
  predictions: length, permutation, weight decomp.; errors: ℓ_1 error, and more

Outlook:

- More sophisticated techniqies to leverage imperfect predictions
- Predictions to improve other performance metrics
  running time (offline alg.), update time (dynamic alg.), price of anarchy (mechanism design), etc.

Summary and Outlook

This talk

- Algorithms with predictions
  binary search, scheduling with unknown job sizes, precedences (online)
- Powerful time-sharing framework
  admits (blackbox) algorithms with error-dependent performance guarantee
- Prediction models and error measures
  predictions: length, permutation, weight decomp.; errors: ℓ_1 error, and more

Outlook:

- More sophisticated techniqies to leverage imperfect predictions
- Predictions to improve other performance metrics
  running time (offline alg.), update time (dynamic alg.), price of anarchy (mechanism design), etc.
- Minimalistic, parsimonious predictions