Competitive Kill-and-Restart and Preemptive Strategies for Non-clairvoyant Scheduling

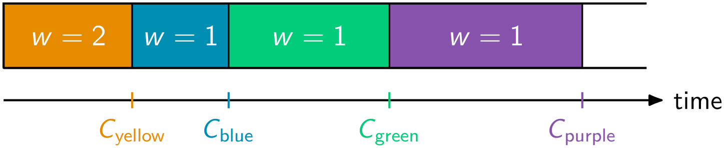
Sven Jäger Guillaume Sagnol Daniel Schmidt genannt Waldschmidt Philipp Warode

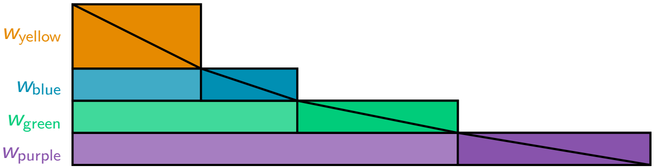
Highlights of Algorithms | 03 June 2023 | Prague

Minimizing the Sum of Weighted Completion Times

Given: Set J of n jobs with processing times $p_j > 0$ and weights $w_j > 0$.

Task: Schedule jobs on a single machine such that the sum of weighted completion times $\sum_{j \in J} w_j C_j$ is minimized.





total area $= \sum_{j \in J} w_j C_j$

minimized if jobs are ordered by Smith ratio p_j / w_j

Preemptive Strategies for Non-clairvoyant Scheduling

What if we don't know the processing times beforehand?

Preemptive Strategies for Non-clairvoyant Scheduling

What if we don't know the processing times beforehand?

- Preemptive scheduling algorithms can interrupt and resume jobs arbitrarily often.
- Cycling over jobs in infinitesimally small times $\longrightarrow$ processing jobs with rates $\in [0, 1]$.

Preemptive Strategies for Non-clairvoyant Scheduling

What if we don't know the processing times beforehand?

- Preemptive scheduling algorithms can interrupt and resume jobs arbitrarily often.
- Cycling over jobs in infinitesimally small times $\longrightarrow$ processing jobs with rates $\in [0,1]$.

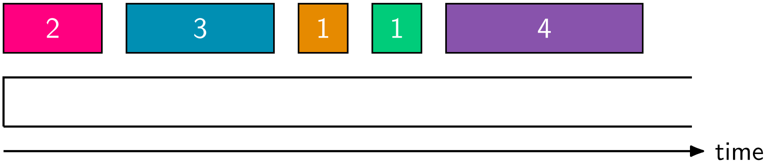
Algorithm: Weighted Round Robin (WRR) / Generalized Processor Sharing (GPS).

At any time t:

1. Let $J(t)$ be the set of unfinished jobs.
2. Process each job $j \in J(t)$ at rate $y_j(t) \leftarrow \frac{w_j}{\sum_{j \in J(t)} w_j}$.

Example:

(unit weights)



Preemptive Strategies for Non-clairvoyant Scheduling

What if we don't know the processing times beforehand?

- Preemptive scheduling algorithms can interrupt and resume jobs arbitrarily often.
- Cycling over jobs in infinitesimally small times $\longrightarrow$ processing jobs with rates $\in [0, 1]$.

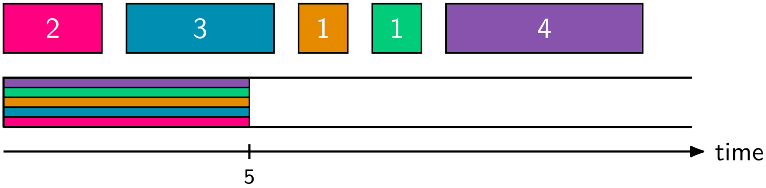
Algorithm: Weighted Round Robin (WRR) / Generalized Processor Sharing (GPS).

At any time t:

1. Let $J(t)$ be the set of unfinished jobs.
2. Process each job $j \in J(t)$ at rate $y_j(t) \leftarrow \frac{w_j}{\sum_{j \in J(t)} w_j}$.

Example:

(unit weights)



Preemptive Strategies for Non-clairvoyant Scheduling

What if we don't know the processing times beforehand?

- Preemptive scheduling algorithms can interrupt and resume jobs arbitrarily often.
- Cycling over jobs in infinitesimally small times $\longrightarrow$ processing jobs with rates $\in [0, 1]$.

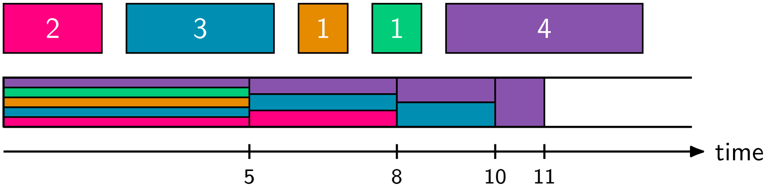
Algorithm: Weighted Round Robin (WRR) / Generalized Processor Sharing (GPS).

At any time t:

1. Let $J(t)$ be the set of unfinished jobs.
2. Process each job $j \in J(t)$ at rate $y_j(t) \leftarrow \frac{w_j}{\sum_{j \in J(t)} w_j}$.

Example:

(unit weights)



Preemptive Strategies for Non-clairvoyant Scheduling

single machine	Kim, Chwa '03: WRR is 2-competitive.
m parallel machines	Motwani et al. '94: If $w = 1$, then RR is 2-competitive.

Preemptive Strategies for Non-clairvoyant Scheduling

	no release dates	with release dates
single machine	Kim, Chwa '03: WRR is 2-competitive.	**Theorem 1.** WSETF *is 2-competitive.*
m parallel machines	Motwani et al. '94: If $w = 1$, then RR is 2-competitive.	

Weighted **S**hortest **E**lapsed **T**ime **F**irst

$\approx$ WRR, but jobs released later catch up

Preemptive Strategies for Non-clairvoyant Scheduling

	no release dates	with release dates
single machine	Kim, Chwa '03: WRR is 2-competitive.	**Theorem 1.** WSETF *is 2-competitive.*
m parallel machines	Motwani et al. '94: If $w = 1$, then RR is 2-competitive.	

Weighted **S**hortest **E**lapsed **T**ime **F**irst

$\approx$ WRR, but jobs released later catch up

Motwani et al. '94: No non-clairvoyant preemptive algorithm can have a competitive ratio better than 2.

Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

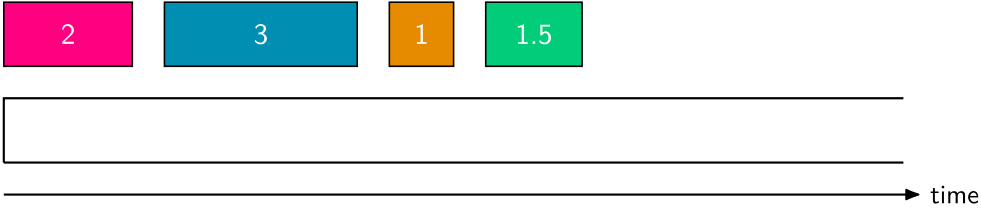
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



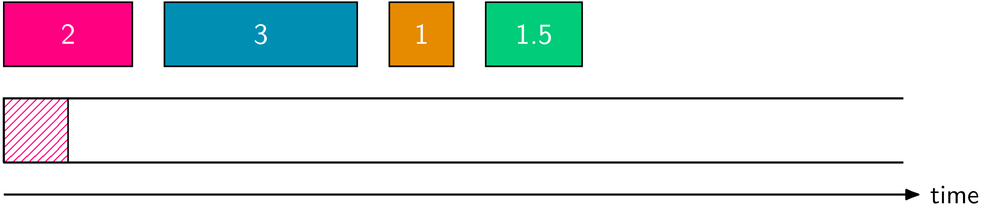
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



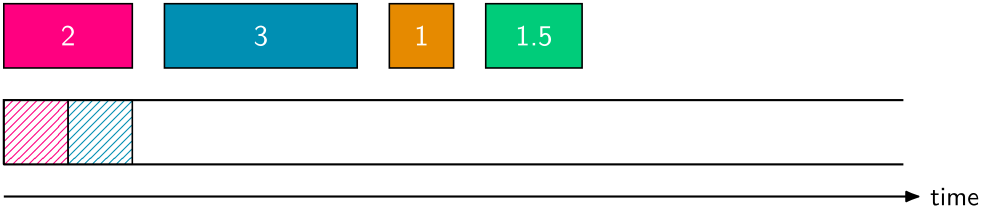
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



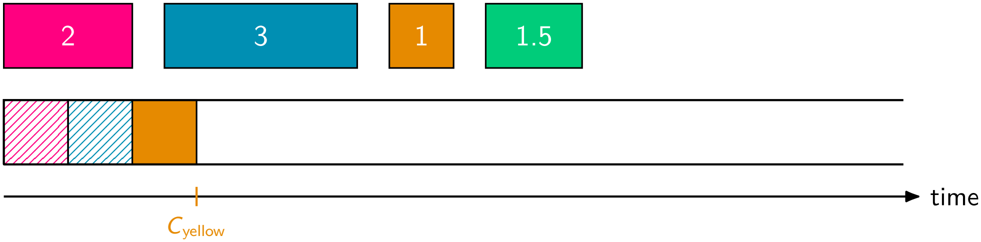
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



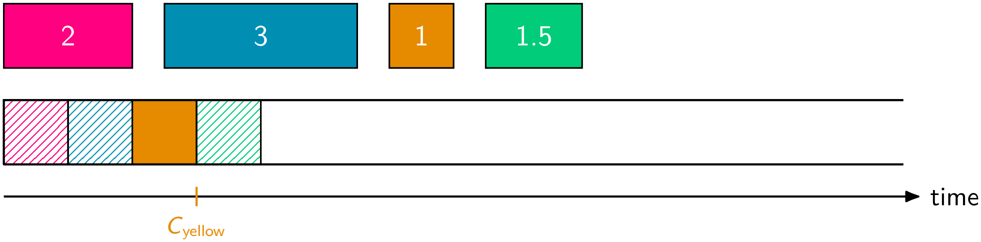
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



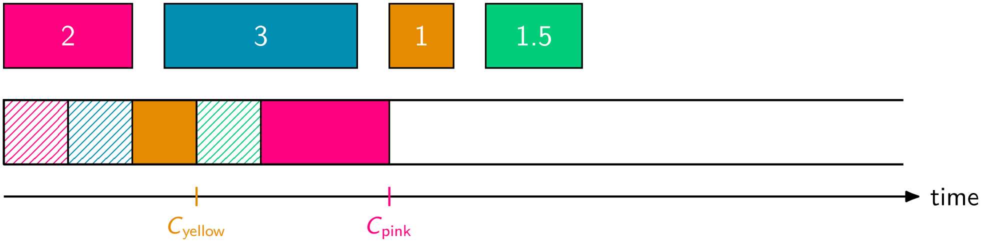
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic *b*-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   - a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   - b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



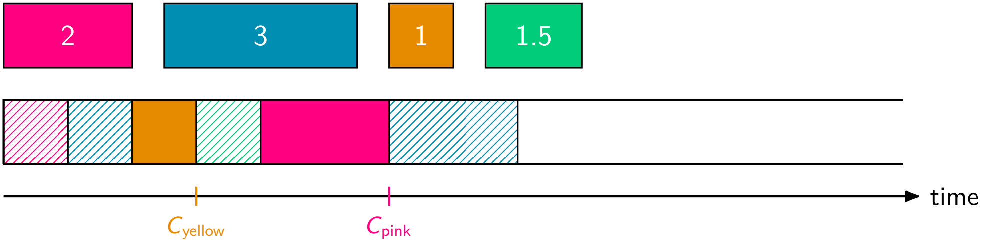
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



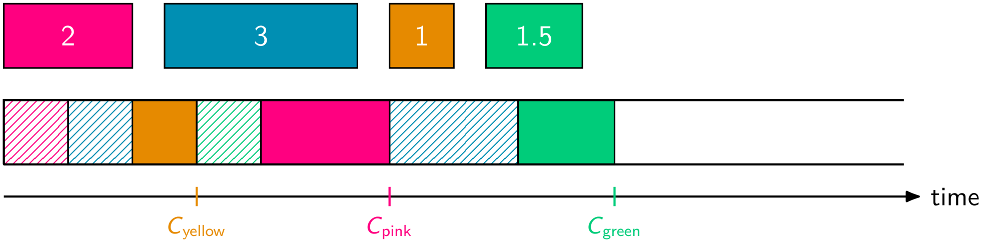
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



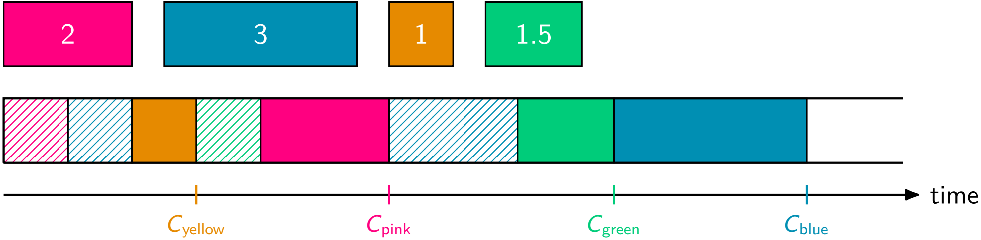
Kill-and-Restart Strategies for Non-clairvoyant Scheduling

A job may be interrupted at any time. Interrupted jobs have to be started from scratch.

Algorithm: Deterministic b-Scaling.

1. Set $i \leftarrow 0$
2. While not all jobs are finished:
   a. Probe all unfinished jobs in a fixed order σ for a time $\tau = w_j b^i$
   b. $i \leftarrow i + 1$

Example ($b = 2$ and unit weights)



Competitiveness of the b-Scaling Strategy

- The 2-competitiveness of WRR implies that the 2-scaling strategy is 8-competitive.
- We refine the analysis and determine the exact competitive ratio of this strategy.

Competitiveness of the b-Scaling Strategy

- The 2-competitiveness of WRR implies that the 2-scaling strategy is 8-competitive.
- We refine the analysis and determine the exact competitive ratio of this strategy.

Theorem 2.
The b-scaling strategy is $\left(1 + \frac{2b^{\frac{3}{2}}}{b-1}\right)$-competitive, and this bound is tight. For $b = 3$ this yields $1 + 3\sqrt{3} \approx 6.196$.

Competitiveness of the b-Scaling Strategy

- The 2-competitiveness of WRR implies that the 2-scaling strategy is 8-competitive.
- We refine the analysis and determine the exact competitive ratio of this strategy.

Theorem 2.
The b-scaling strategy is $\left(1 + \frac{2b^{\frac{3}{2}}}{b-1}\right)$*-competitive, and this bound is tight. For* $b = 3$ *this yields* $1 + 3\sqrt{3} \approx 6.196$.

Randomization: Choose random permutation and random probing offset.

Theorem 3.
The rand. b-scaling strategy is $\frac{\sqrt{b}+2b-1}{\sqrt{b}\ln b}$*-competitive, and this bound is tight. For* $b \approx 8.16$ *this yields* ≈ 3.032.

Kill-and-Restart Strategies for Non-clairvoyant Scheduling

	no release dates	with release dates
single machine	**Theorem 2.** *b-scaling is* 6.196*-competitive (and not better).*	**Theorem 5.** *b-scaling is* 9.915*-competitive.*
m parallel machines	**Theorem 6.** *If* $w = 1$*, then b-scaling is* 9.899*-competitive.*	

Kill-and-Restart Strategies for Non-clairvoyant Scheduling

	no release dates	with release dates
single machine	**Theorem 2.** *b-scaling is* 6.196*-competitive (and not better).*	**Theorem 5.** *b-scaling is* 9.915*-competitive.*
m parallel machines	**Theorem 6.** *If* $w = 1$*, then b-scaling is* 9.899*-competitive.*	

Theorem 7.
No deterministic non-clairvoyant kill-and-restart strategy has competitive ratio better than 3.

Kill-and-Restart Strategies for Non-clairvoyant Scheduling

	no release dates	with release dates
single machine	**Theorem 2.** *b-scaling is* 6.196*-competitive (and not better).*	**Theorem 5.** *b-scaling is* 9.915*-competitive.*
m parallel machines	**Theorem 6.** *If* $w = 1$*, then b-scaling is* 9.899*-competitive.*	

Theorem 7.
No deterministic non-clairvoyant kill-and-restart strategy has competitive ratio better than 3.

See you at the poster!