technische universität
dortmund

Optimal Online Load Maximization with Commitment

Samin Jamalabadi, Uwe Schwiegelshohn
TU Dortmund University
Chris Schwiegelshohn
Sapienza University of Rome

technische universität dortmund

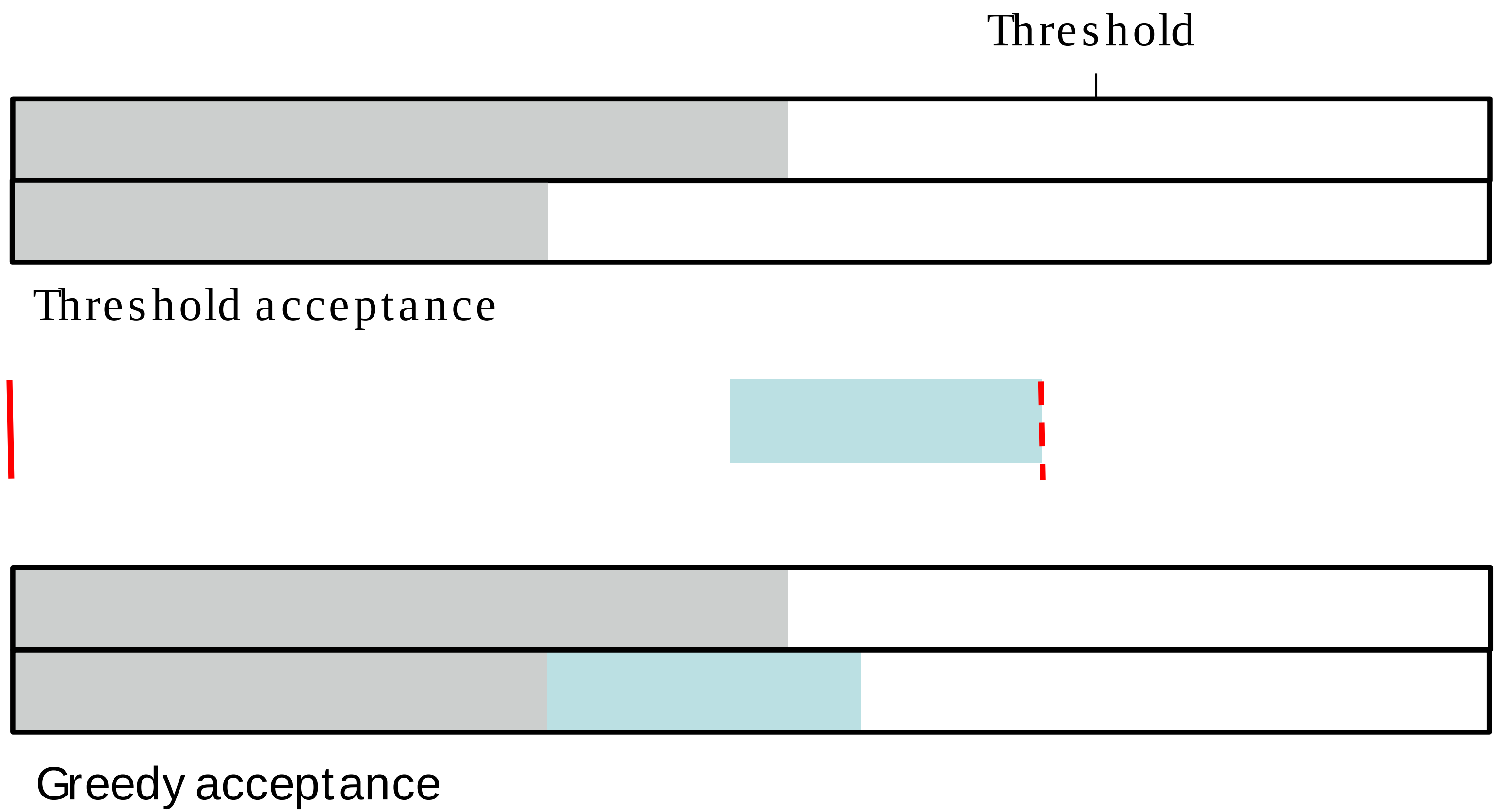
$Pm|online, \varepsilon, commit|\sum p_j \cdot (1 - U_j)$

- Pm: m parallel identical machines.
- $online$: jobs are submitted over time.
  - We do not know the existence nor any properties of future jobs.
- ε: a job J_j has deadline $d_j \geq r_j + \varepsilon \cdot p_j$ with constant slack parameter ε.
  - r_j: submission time of job J_j
  - p_j: processing time of job J_j
- $commit$: we must decide immediately after submission whether to reject a new job J_j ($U_j = 1$) or to accept it ($U_j = 0$).
  - For $U_j = 0$, we must also immediately fix the start time of the job.
  - We must complete every accepted job on time.
- $\sum p_j \cdot (1 - U_j)$: we want to maximize the total processing time of all accepted jobs.

Algorithm Choices

- Acceptance
- Allocation
- Timing

Acceptance Algorithms for $P2$



Algorithm Choices

- Acceptance
  - Greedy: the resulting schedule completes all accepted jobs on time.
  - Threshold: the deadline of an accepted job is at least as large as a deadline threshold.
- Allocation
- Timing

Allocation for $P2$ with Greedy Acceptance

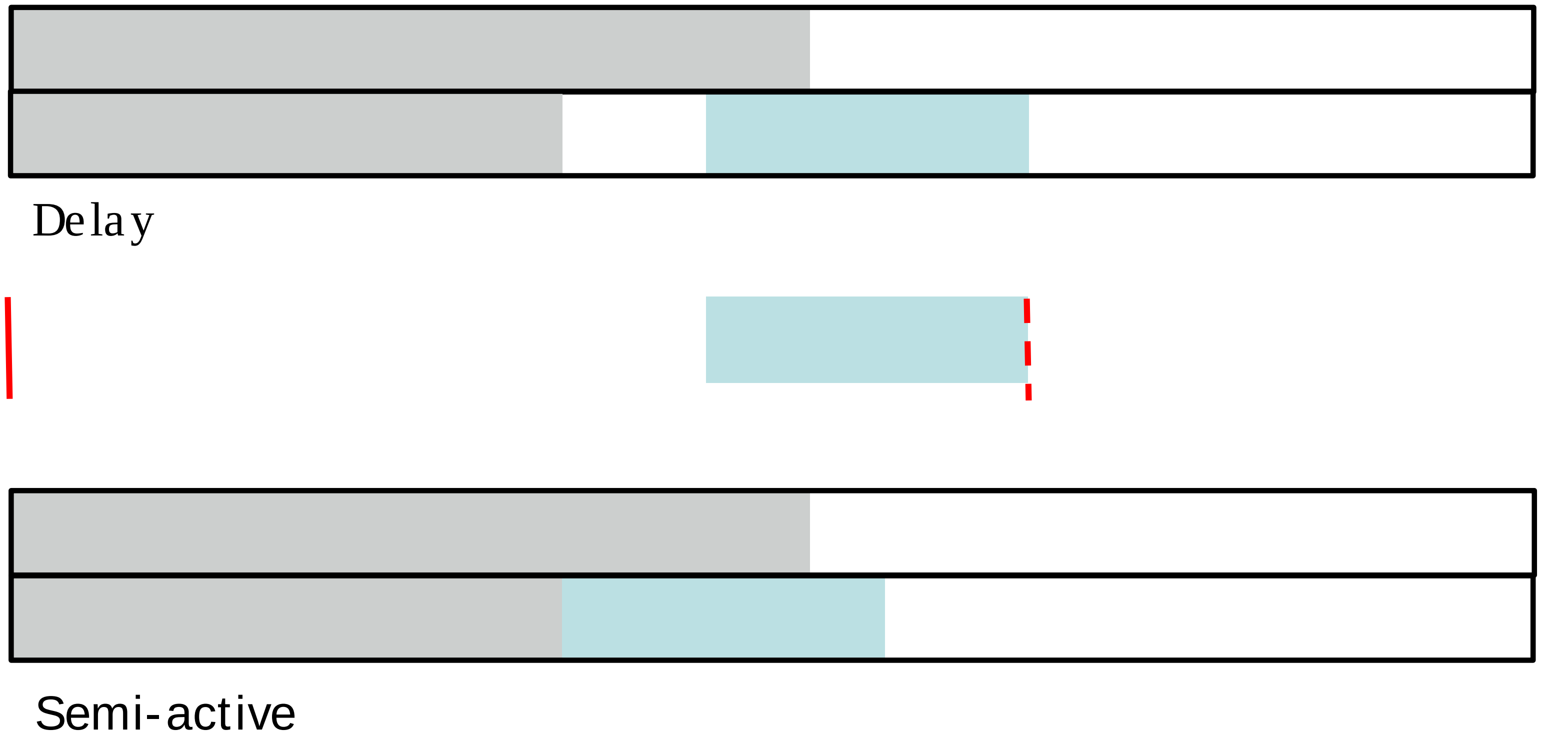
Skewed allocation

Balanced allocation

Algorithm Choices

- Acceptance
  - Greedy: the resulting schedule completes all accepted jobs on time.
  - Threshold: the deadline of an accepted job is at least as large as a deadline threshold.
- Allocation
  - Skewed: the candidate machine with the highest load
  - Balanced: the candidate machine with the minimal load (for greedy) or the candidate machine that increases the threshold by the smallest amount (for threshold).
- Timing

Job Starting for $P2$ with Greedy Acceptance



Algorithm Choices

- Acceptance
  - Greedy: the resulting schedule completes all accepted jobs on time.
  - Threshold: the deadline of an accepted job is at least as large as a deadline threshold.
- Allocation
  - Skewed: the candidate machine with the highest load
  - Balanced: the candidate machine with the minimal load (for greedy) or the candidate machine that increases the threshold by the smallest amount (for threshold).
- Timing
  - Semi-active: as early as possible on the allocated machine.
  - Delay: possible intermediate idle time on the allocated machine.

Threshold Calculation

- The machines are indexed in decreasing order of their outstanding loads.
- For machine m_i, we calculate a machine specific threshold using a function $f_i(\varepsilon)$ and the outstanding load of the machine at time t.

$$d_{lim,i}\Big|_t = load(m_i)\Big|_t \cdot f_i(\varepsilon) + t$$

- The threshold is the maximum of the machine specific thresholds.

$$d_{lim}\Big|_t = \max_{1 \le i \le m} d_{lim,i}\Big|_t$$

- We set $f_m(\varepsilon) = \frac{1+\varepsilon}{\varepsilon}$ and determine the remaining $f_i(\varepsilon)$ recursively.

$$\frac{m \cdot f_i(\varepsilon) + 1}{\sum_{h=1}^{i-1} f_i(\varepsilon) - (i-1) + 1} = const \ \text{ for } \ 1 \le i \le m$$

Competitive Ratio

- Threshold allocation with a skewed and semi-active schedule has the competitive ratio
$$m \cdot f_1(\varepsilon) + 1 \geq 2m + 1 \ \text{ for } f_1(\varepsilon) \geq 2.$$
- $\varepsilon_T = \arg\max\{f_1(\varepsilon) = 2\}$ decreases with increasing m.

m	2	3	4	5
ε_T	0.2857	0.0900	0.0291	0.0098

- Greedy allocation with a skewed and semi-active schedule has the competitive ratio
$$\frac{1}{m} + \frac{1+\varepsilon}{\varepsilon} \ \text{ for } \ 0 < \varepsilon \leq 1.$$

Greedy Acceptance

- Intuitively, greedy acceptance is the simplest approach.

- The competitive ratio of greedy acceptance is identical to the competitive ratio of the following min-threshold approach for $0 < \varepsilon \leq 1$:

$$d_{lim,i}\Big|_t = load(m_i)\Big|_t \cdot \frac{1+\varepsilon}{\varepsilon} + t$$

$$d_{lim}|_t = \min_{1 \leq i \leq m} d_{lim,i}\Big|_t$$

Proof Concepts

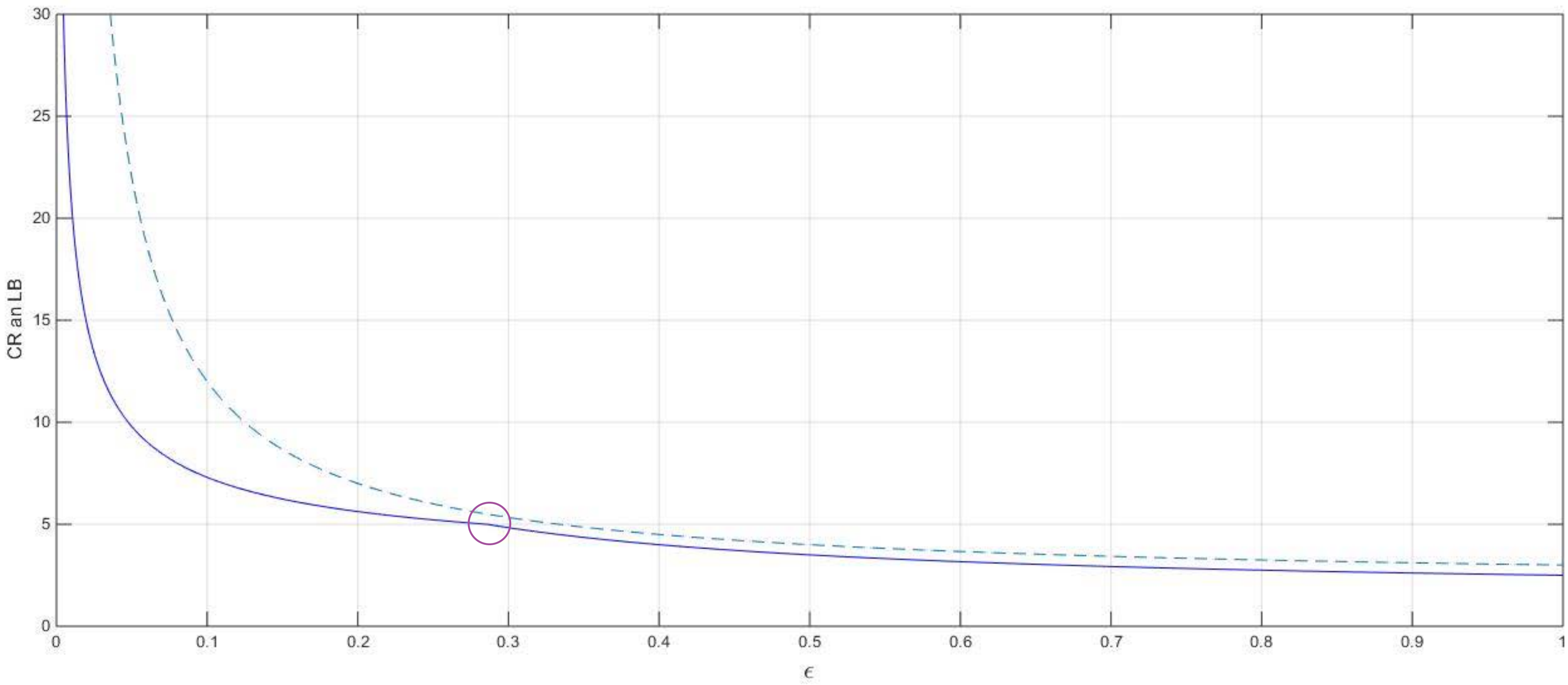
- Key lemma for the (max-)threshold approach:
  - Allocation of a new job to a machine without the maximum outstanding load will turn this machine into the machine with the maximum outstanding load if $f_1(\varepsilon) \geq 2$ holds.

- Partitioning of the resulting schedule into several intervals
  - We determine how much load of every interval cannot be executed outside of this interval in any optimal schedule that has accepted the corresponding jobs.

Previous Results

- For $m = 1$, greedy acceptance (Goldwasser 1999, 2003) with a semi-active schedule has the tight competitive ratio
$$1 + \frac{1+\varepsilon}{\varepsilon} \text{ for } 0 < \varepsilon.$$
- For $m > 1$, greedy allocation with a balanced and semi-active schedule has the competitive ratio (Kim, Chwa 2001)
$$1 + \frac{1+\varepsilon}{\varepsilon} \text{ for } 0 < \varepsilon.$$

Results for $P2$



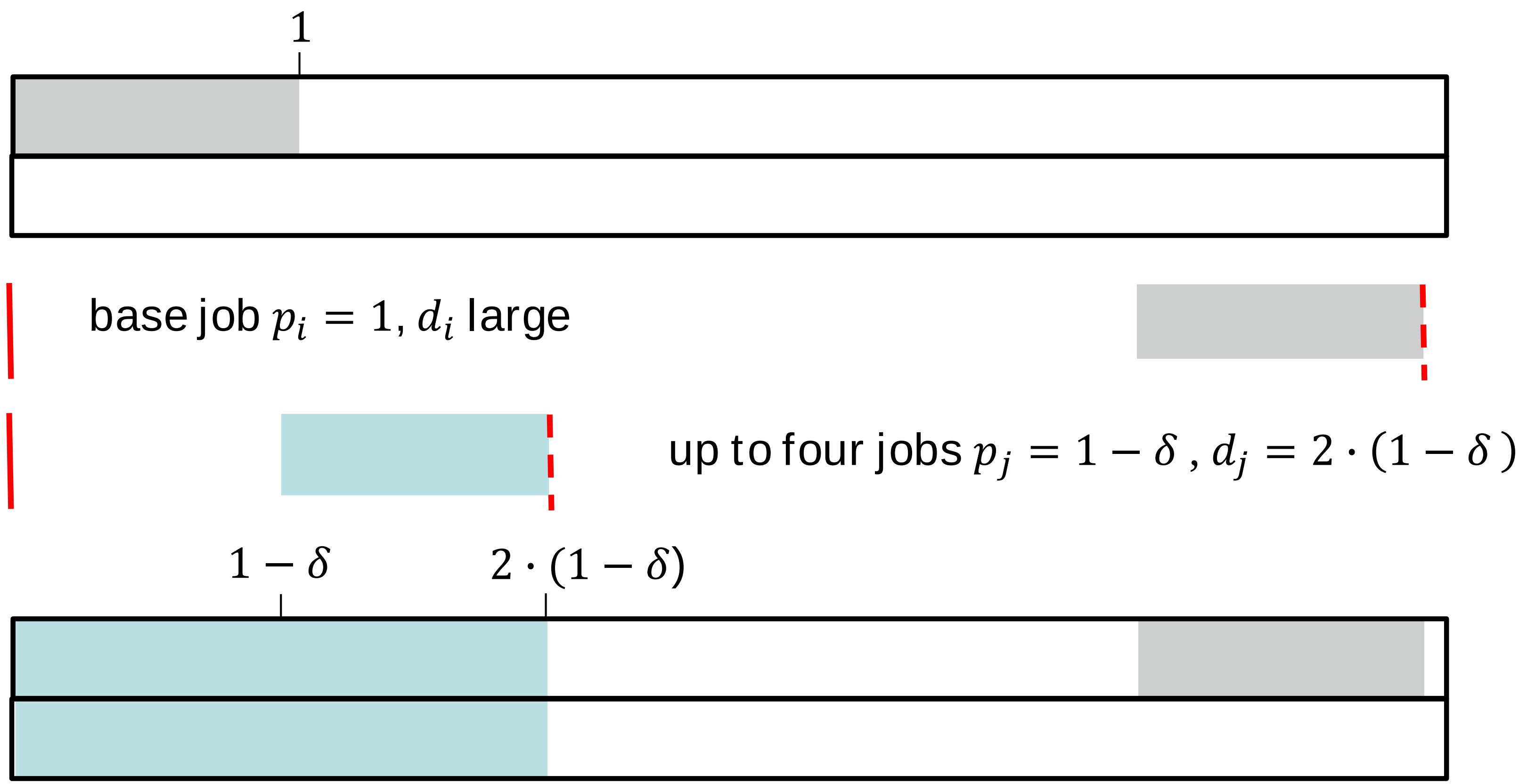
Competitive Ratio

- Gap between greedy and (max-)threshold allocation at ε_T

m	2	3	4	5
Gap	0	5.44	26.61	92.24

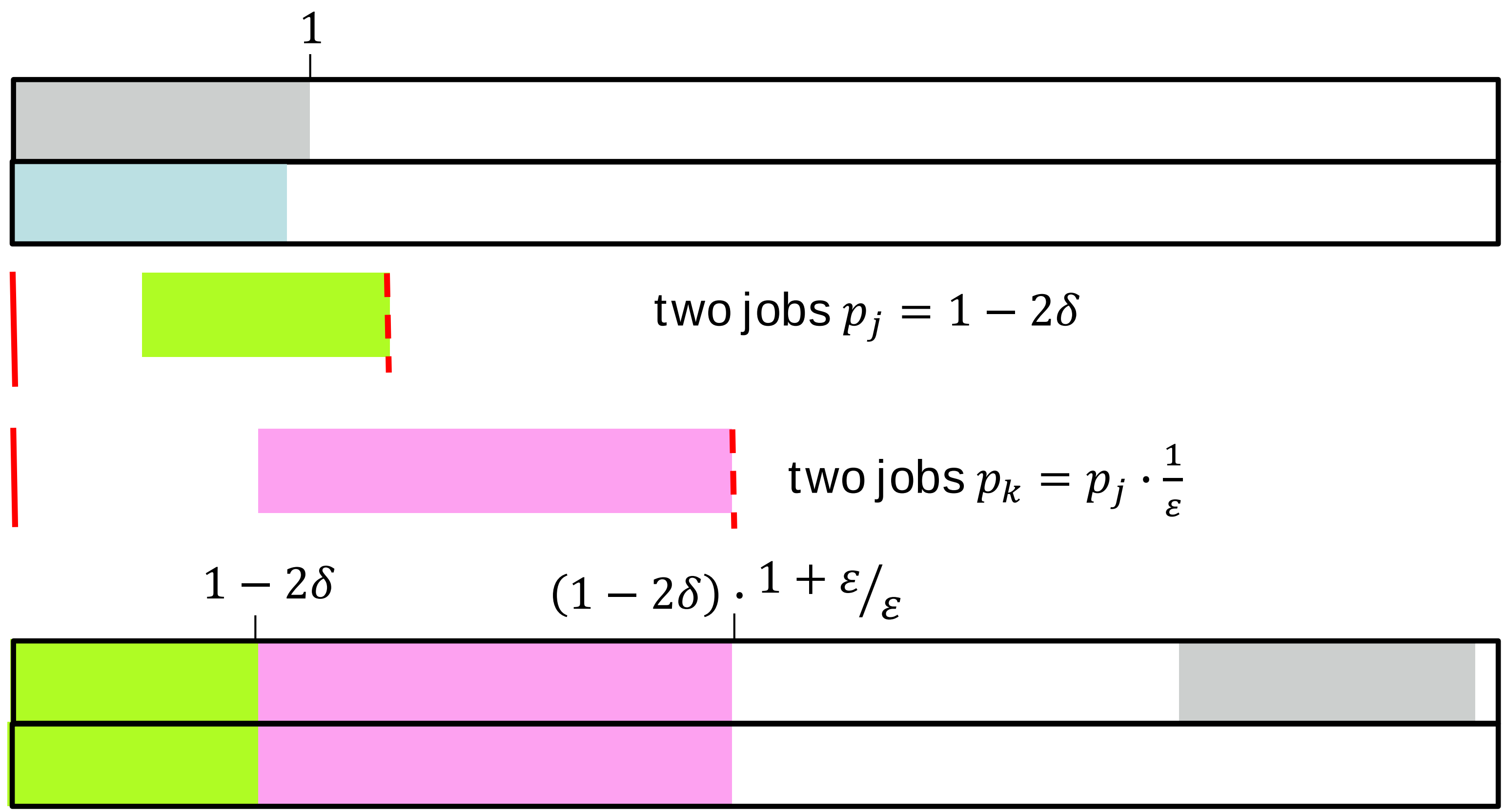
- For $m > 2$, we need m algorithms covering different intervals within $(0,1]$.
- The algorithms are a combination of min-thresholds and max-thresholds.

Lower Bound for $P2$ and Greedy Allocation



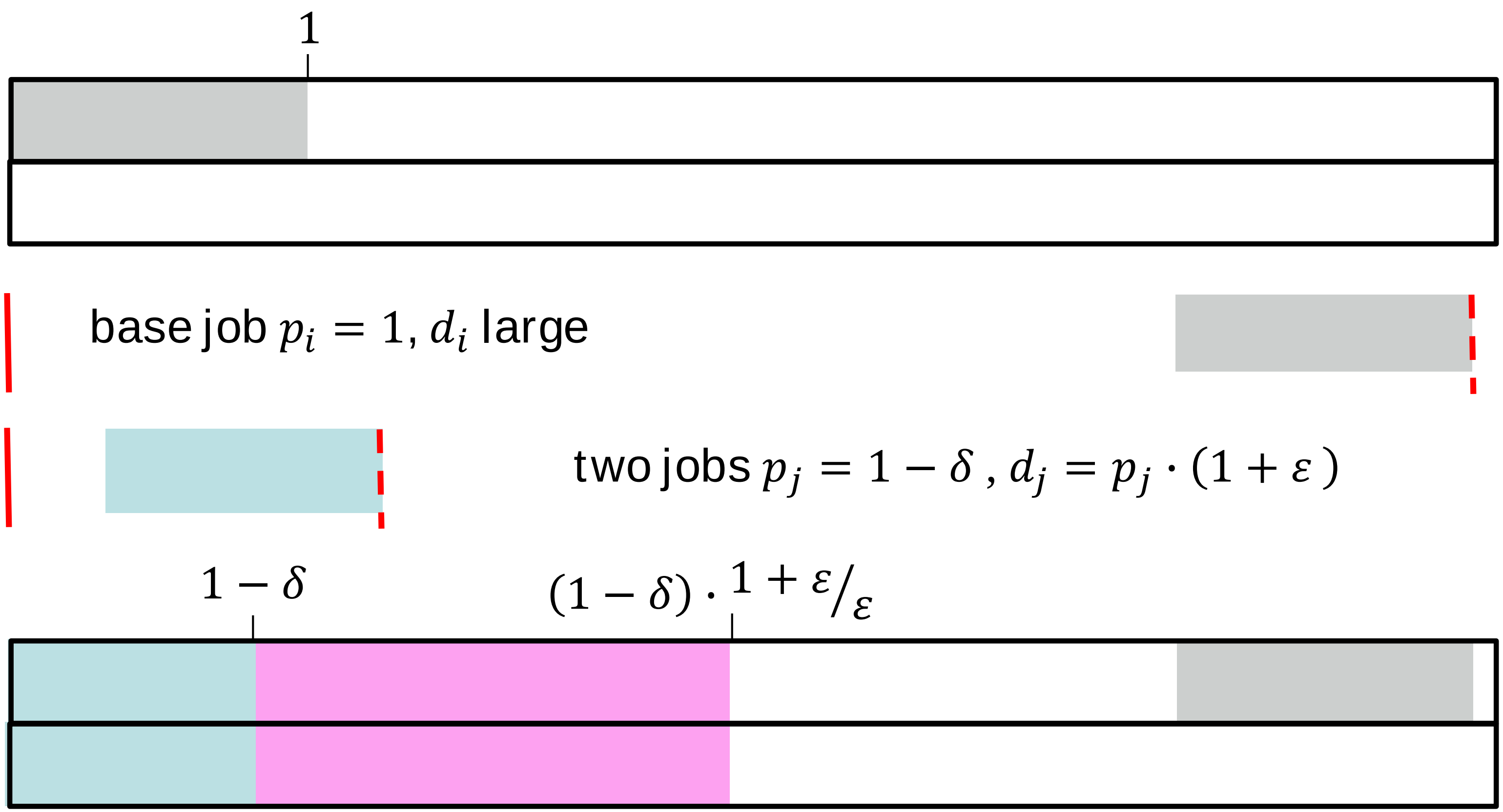
optimal schedule $c = 5$

Lower Bound for $P2$ and Greedy Allocation



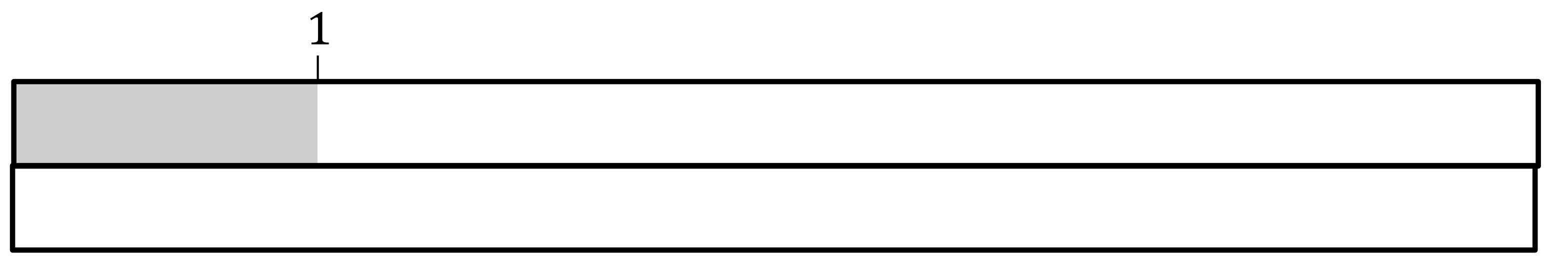
optimal schedule $c = (1 + 2 \cdot {}^{1+\varepsilon}/_{\varepsilon})/2$

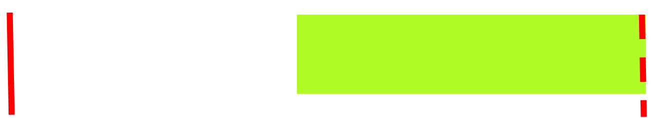
Lower Bound for $P2$ and (Max-)Threshold Allocation



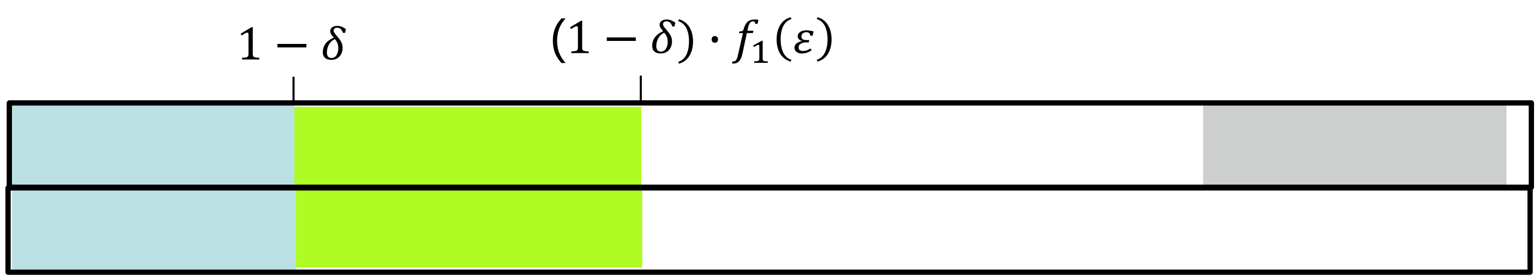
optimal schedule $c = (1 + 2 \cdot {}^{1+\varepsilon}/_{\varepsilon})/2$

Lower Bound for $P2$ and (Max-)Threshold Allocation



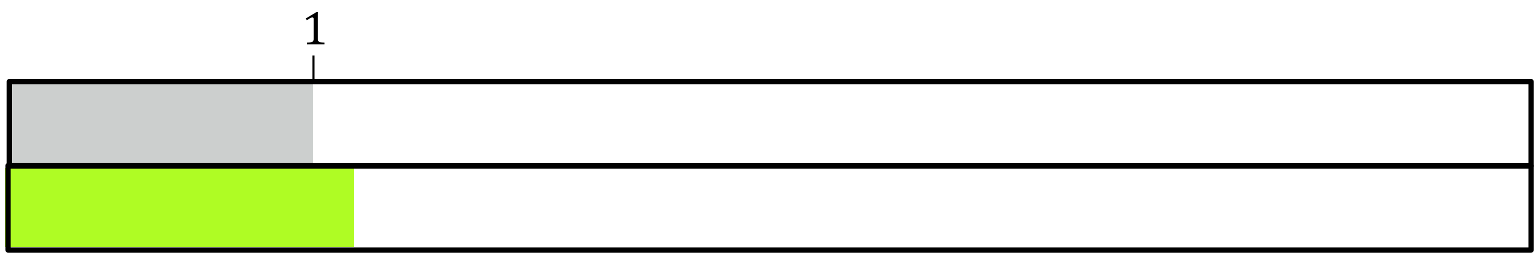


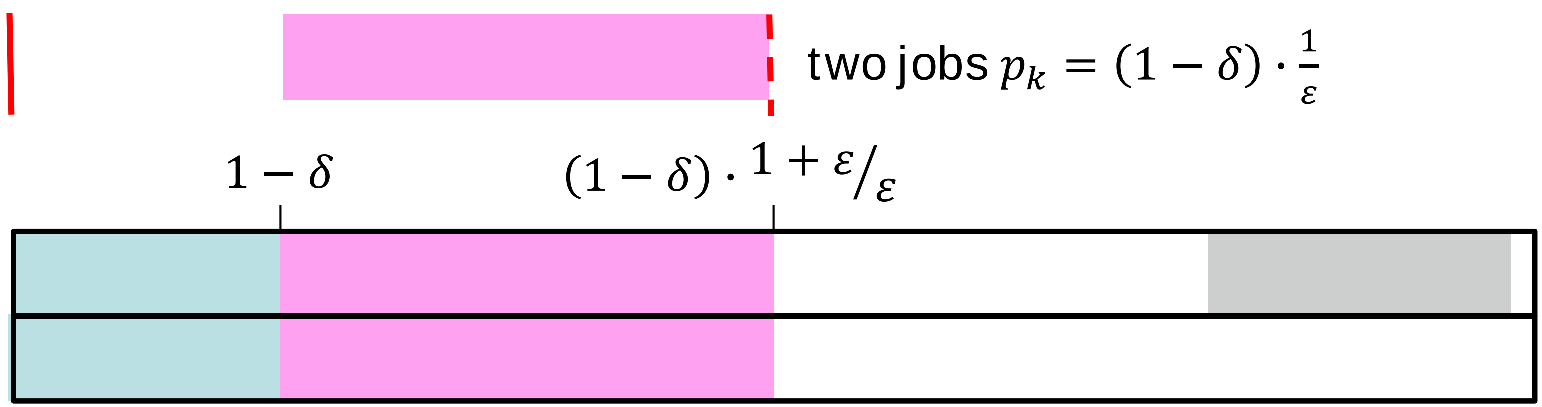
up to two jobs $p_j = (1 - \delta) \cdot (f_1(\varepsilon) - 1)$,
$d_j = (1 - \delta) \cdot f_1(\varepsilon)$



optimal schedule $c = 1 + 2f_1(\varepsilon)$

Lower Bound for $P2$ and (Max-)Threshold Allocation



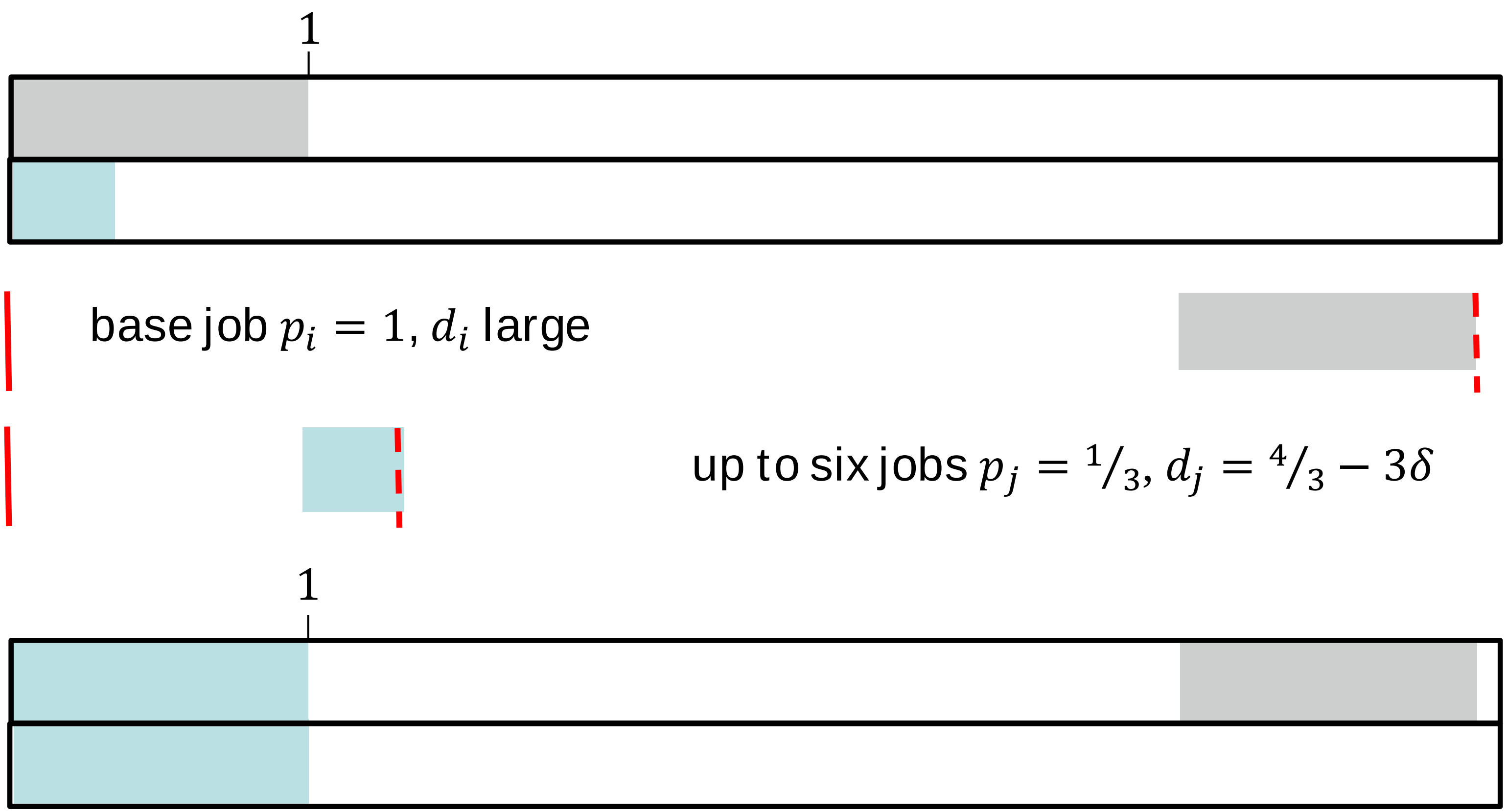


optimal schedule $c = (1 + 2\cdot {}^{1+\varepsilon}/_{\varepsilon})/f_1(\varepsilon) = 1 + 2f_1(\varepsilon)$

Interval $(1, \infty)$ for ε

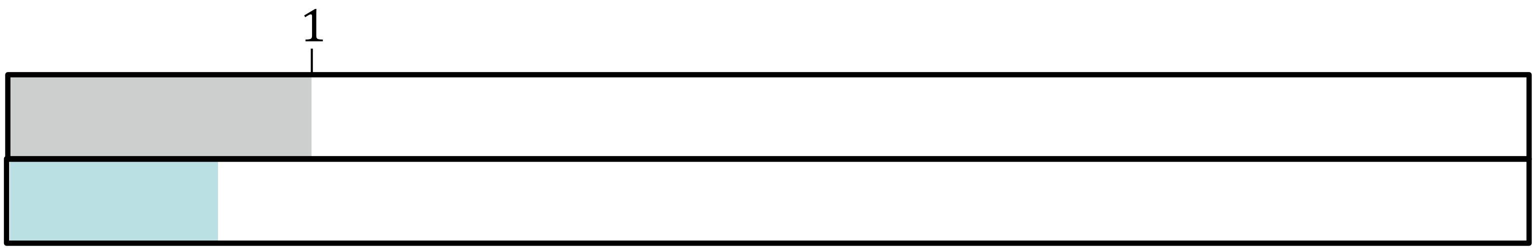
- The optimal competitive ratio is larger for any $\varepsilon \in (0,1]$ than the optimal competitive ratio for any $\varepsilon \in (1, \infty)$.
  - The problem becomes easier for $\varepsilon > 1$?
  - Previous results seem to support this claim.
- Observation: The presented optimal online algorithms for $\varepsilon \in (0,1]$ only use semi-active schedules avoiding any start-time problem
- It is not possible to obtain the competitive ratio $\frac{1}{m} + \frac{1+\varepsilon}{\varepsilon}$ for all $\varepsilon \in (1, \infty)$ when using only semi-active schedules.
  - We consider an example with $\varepsilon = 2$ and the $P2$ environment.
- Progression of time limits the competitive ratio for large ε and $m \geq 3$.

Lower Bound for $P2$, $\varepsilon = 2$, and Semi-active Schedules

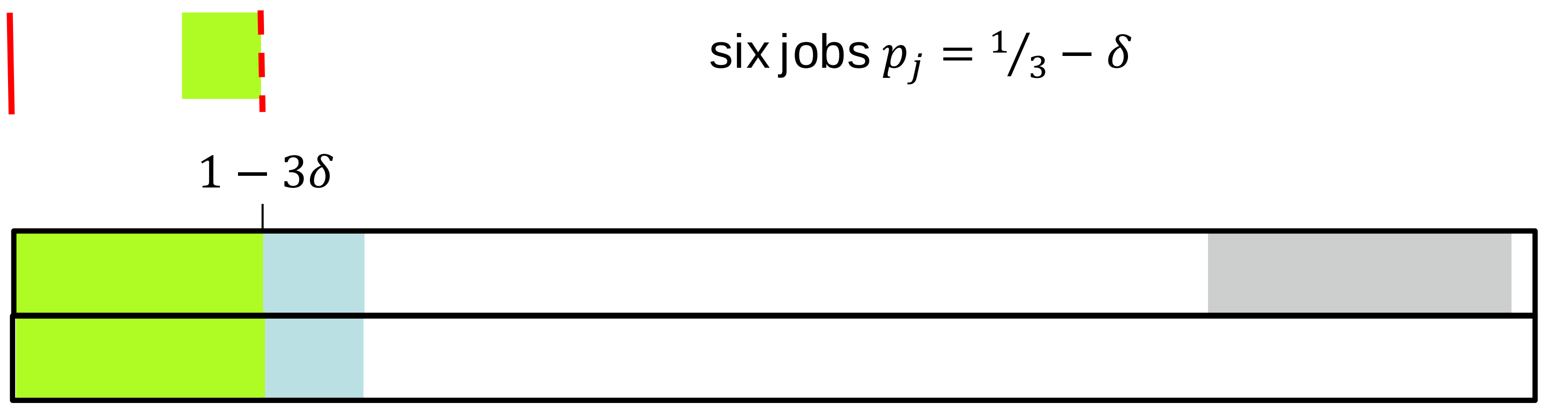


optimal schedule $c = {}^9\!/_4$

Lower Bound for $P2$, $\varepsilon = 2$, and Semi-active Schedules

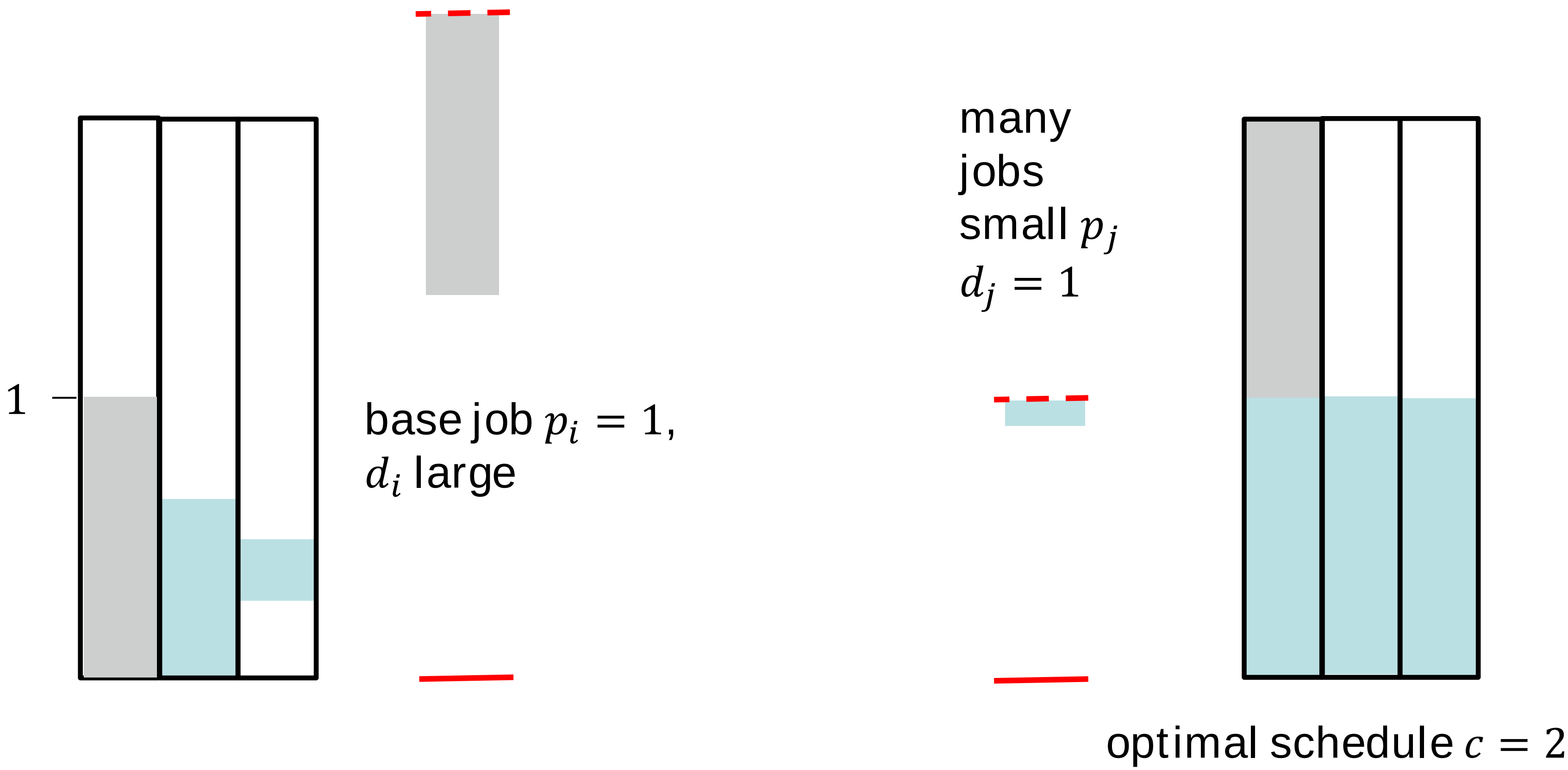


six jobs $p_j = {}^1/_3 - \delta$

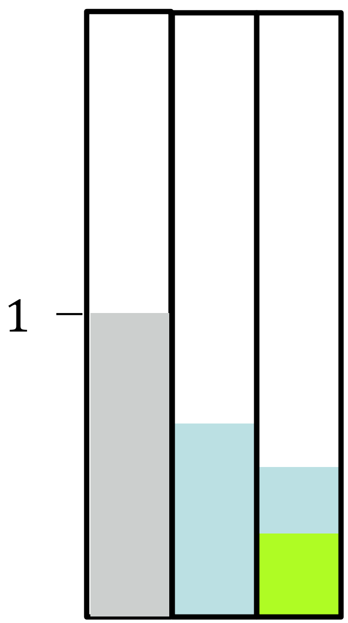


optimal schedule $c = {}^{11}/_5$

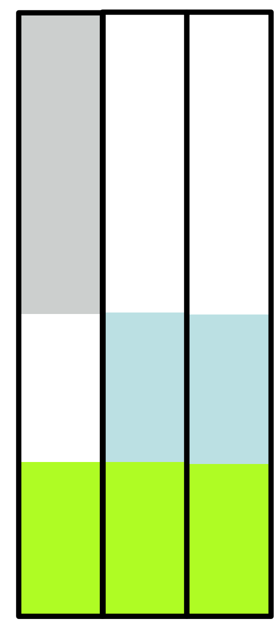
Lower Bound for Pm and Large ε



Lower Bound for Pm and Large ε



many
jobs
small p_j
$d_j = {}^1/_2$



optimal schedule $c = {}^{4m+2}/_{3m+1}$

Conclusion

- For the problem $Pm|online, \varepsilon, commit|\sum p_j \cdot (1-U_j)$, we presented online algorithms with an optimal competitive ratio for $0 < \varepsilon \leq 1$.
- The optimal algorithm consists of m different algorithms that are each valid for a subinterval of $(0,1]$ of ε.
- The algorithms use thresholds, skewed allocation and semi-active schedules.
- For $\varepsilon > 1$, the optimal competitive ratio is smaller but the algorithms are more complicated.