technische universität
dortmund

Parallel Preemptive Online Scheduling with Deadlines and Slack

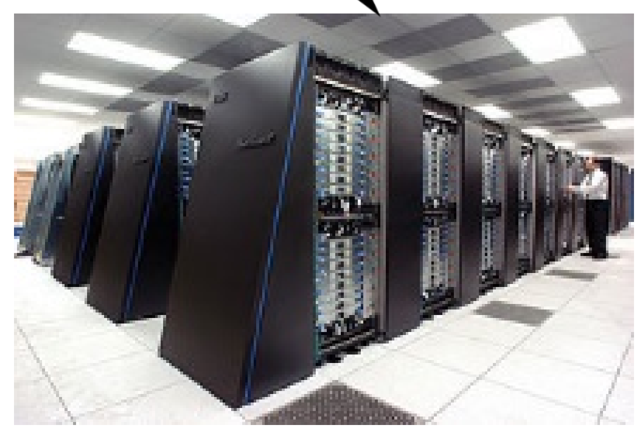
Uwe Schwiegelshohn
TU Dortmund University

A Basic Online Problem in a Large Computer System

Can I have a virtual machine for one hour?

Sure

Binding contract!

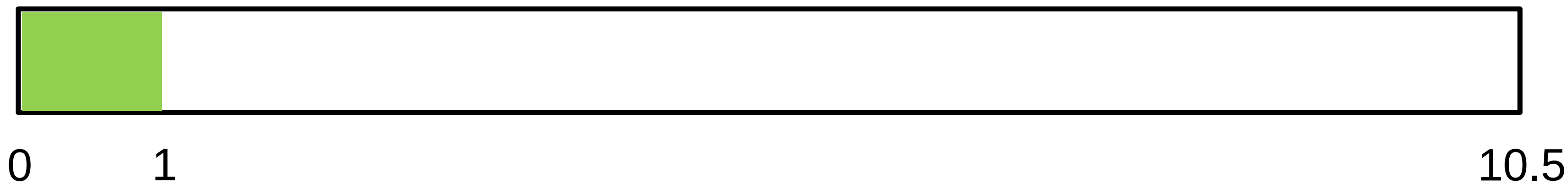


Customer

Provider

Dilemma of the Provider

Known request: utilization 1



Potential future request: utilization 10

Reject of future request due to contract obligation

Competitive ratio of $\sum p_j \cdot (1 - U_j)$ is unbounded!

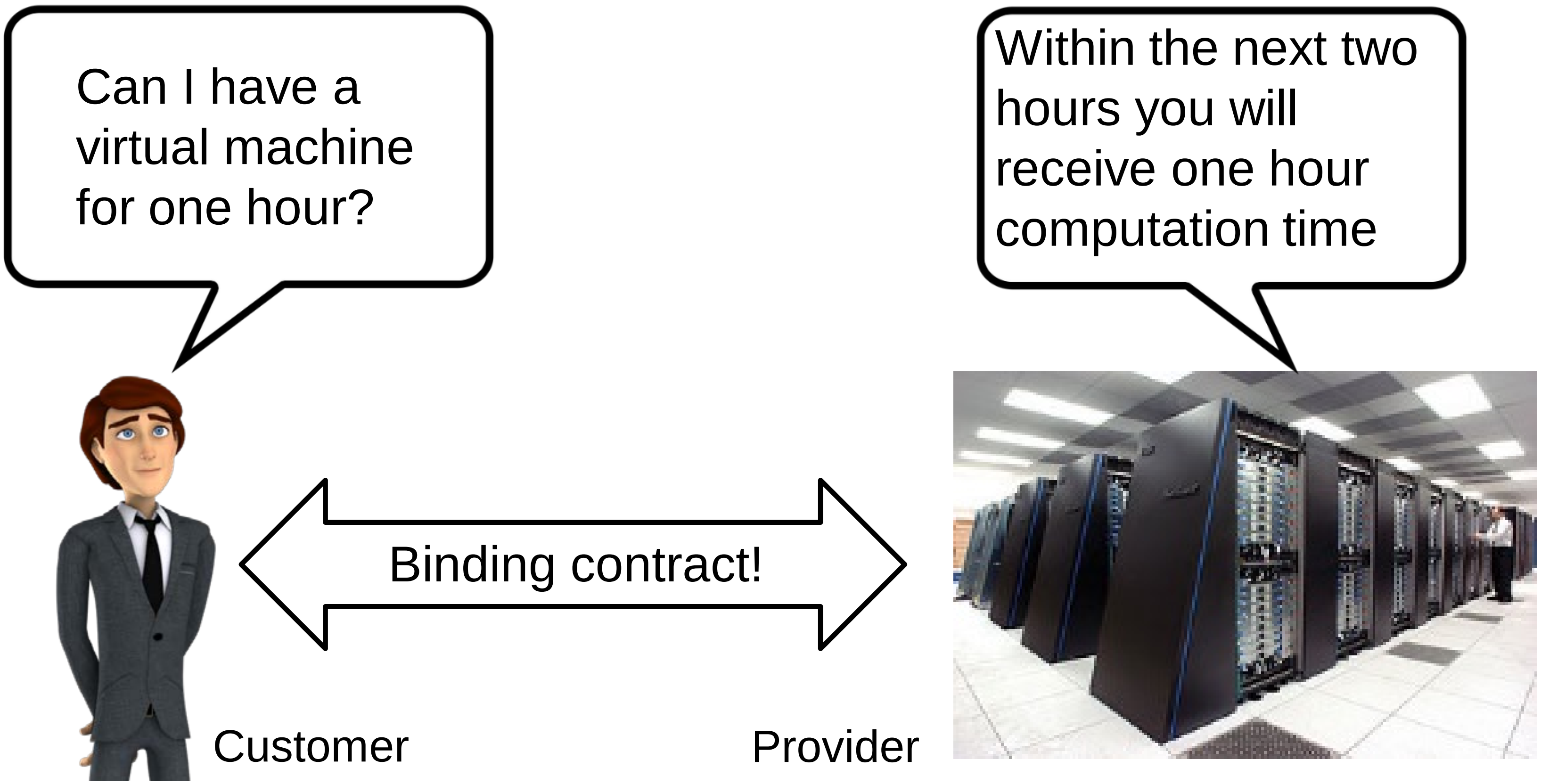
acceptance indicator 0/1

technische universität
dortmund

Competitive Ratio

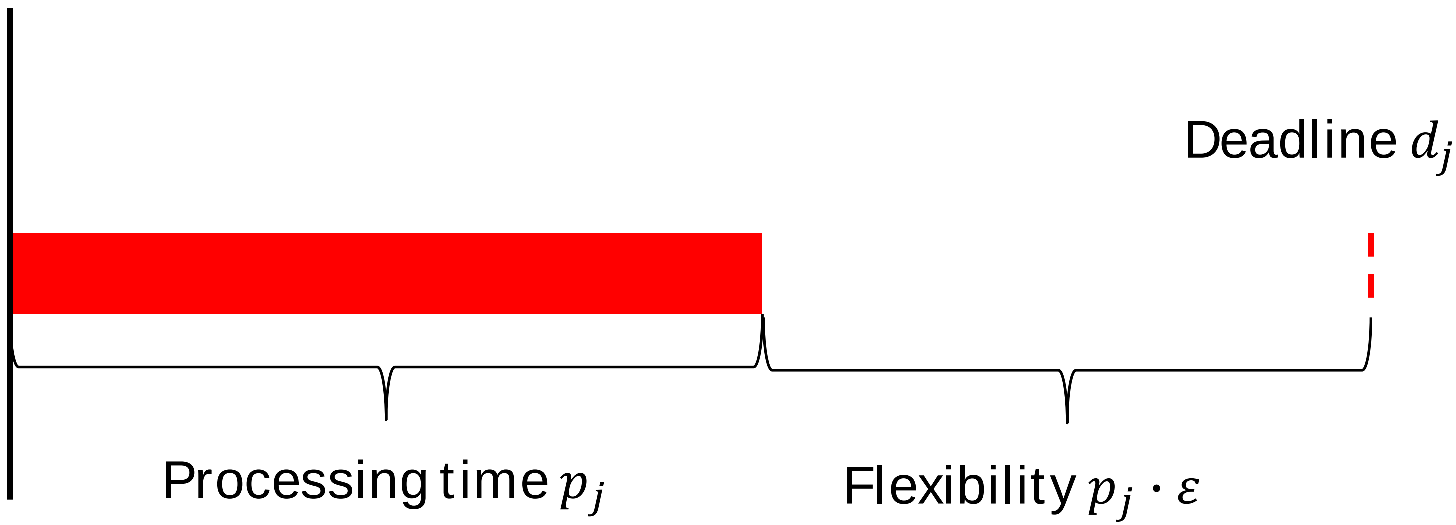
- Worst case analysis of online algorithms: competitive analysis
  - Competitive ratio of algorithm Alg: $\max_{Instances\ I} \frac{\sum p_j \cdot \left(1-U_j(Alg)\right)}{\sum p_j \cdot \left(1-U_j(OPT)\right)}$
- An online algorithm is optimal if the competitive ratio of the algorithm matches the lower bound of the competitive ratio for the online problem.

A Little Flexibility: the Slack



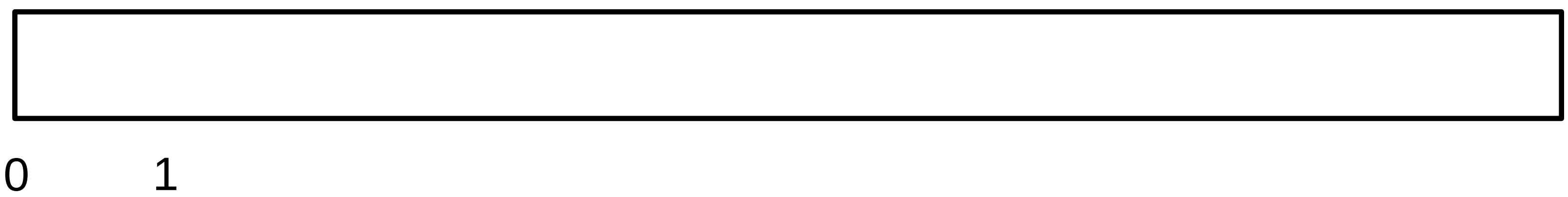
Formal Definition of Slack ε

Release date r_j or submission time = reference time t_{ref}

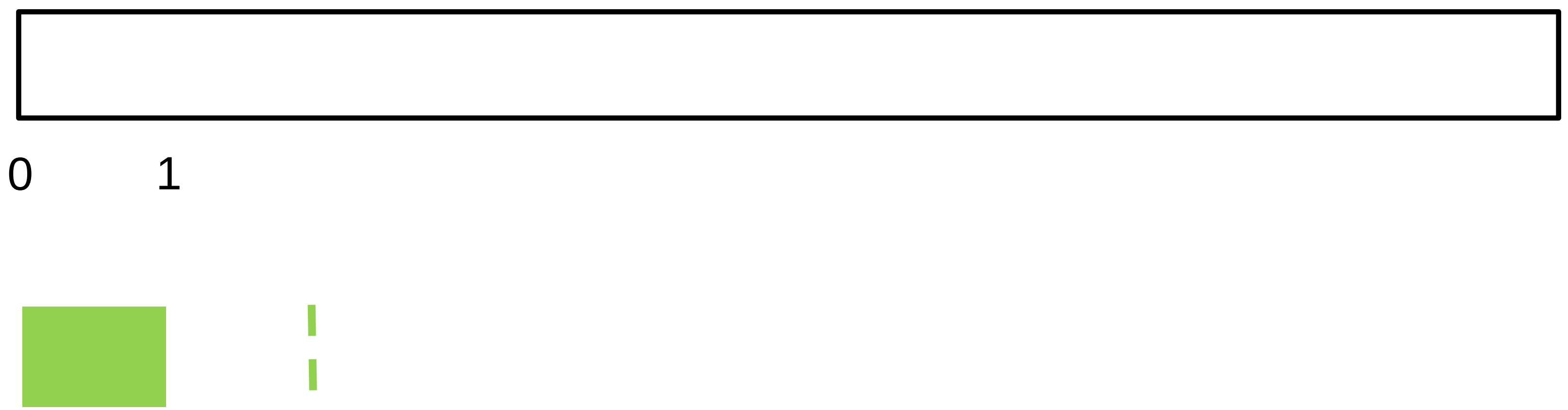


$$d_j \geq r_j + p_j \cdot (1 + \varepsilon)$$

The Benefit of the Slack



The Benefit of the Slack



technische universität
dortmund

The Benefit of the Slack

The Benefit of the Slack

All contracts are valid!

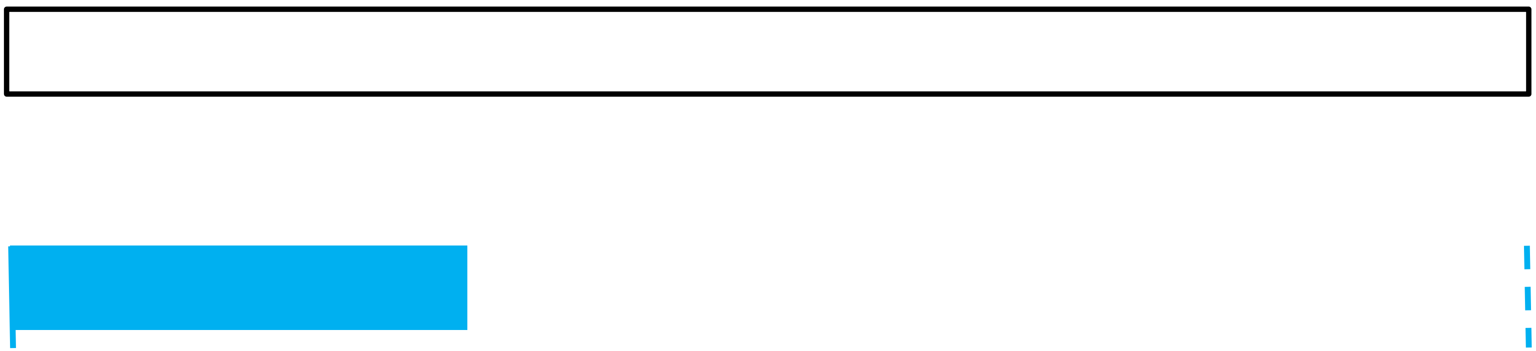
Content of this Talk

- Known Results
- Greedy Acceptance Strategy: If you can accept it, execute it!
- A Lazy Acceptance Strategy: Do not accept all jobs that you can accept!
- The Sequence Problem: All jobs arrive at time 0 in a sequence.
- Lower Bound: The game of the adversary
- Progression of Time: It is quite a difference!
- Restrictions in Practice

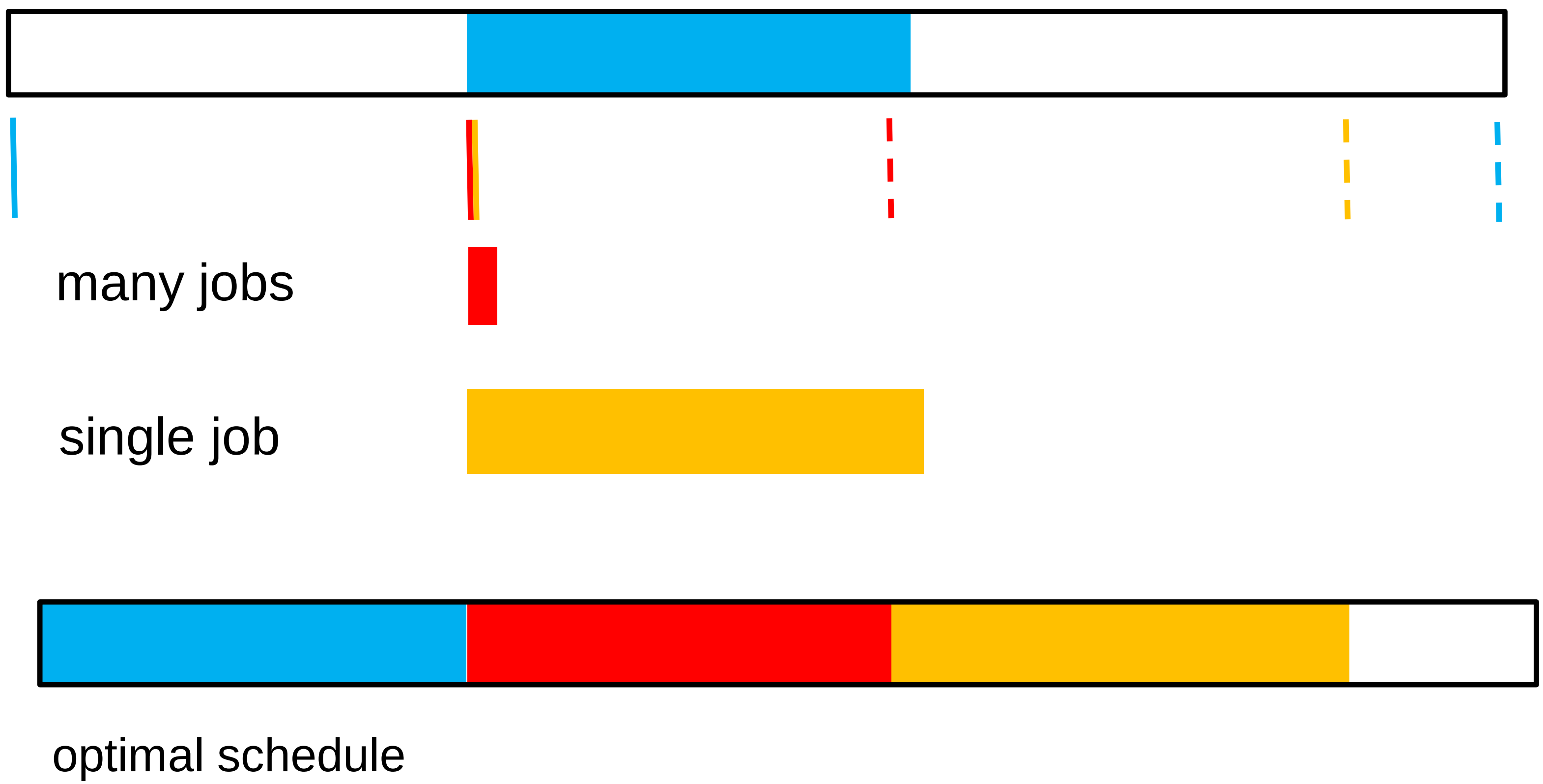
Known Results without Preemptions

- Single machine without preemptions
  - Greedy is $2 + {}^{1}/_{\varepsilon}$ competitive. The algorithm is optimal (Goldwasser 1999).
- Parallel identical machines without preemption
  - Transfer of the single machine algorithm to parallel identical machines (Kim and Chwa 2001).
  - No lower bound is known.
  - Lee (2003) suggested an algorithm with a claimed competitive ratio $m + 1 + m \cdot \varepsilon^{-1/m}$. The real competitive ratio is much larger.

Single Machine Bound without Preemptions



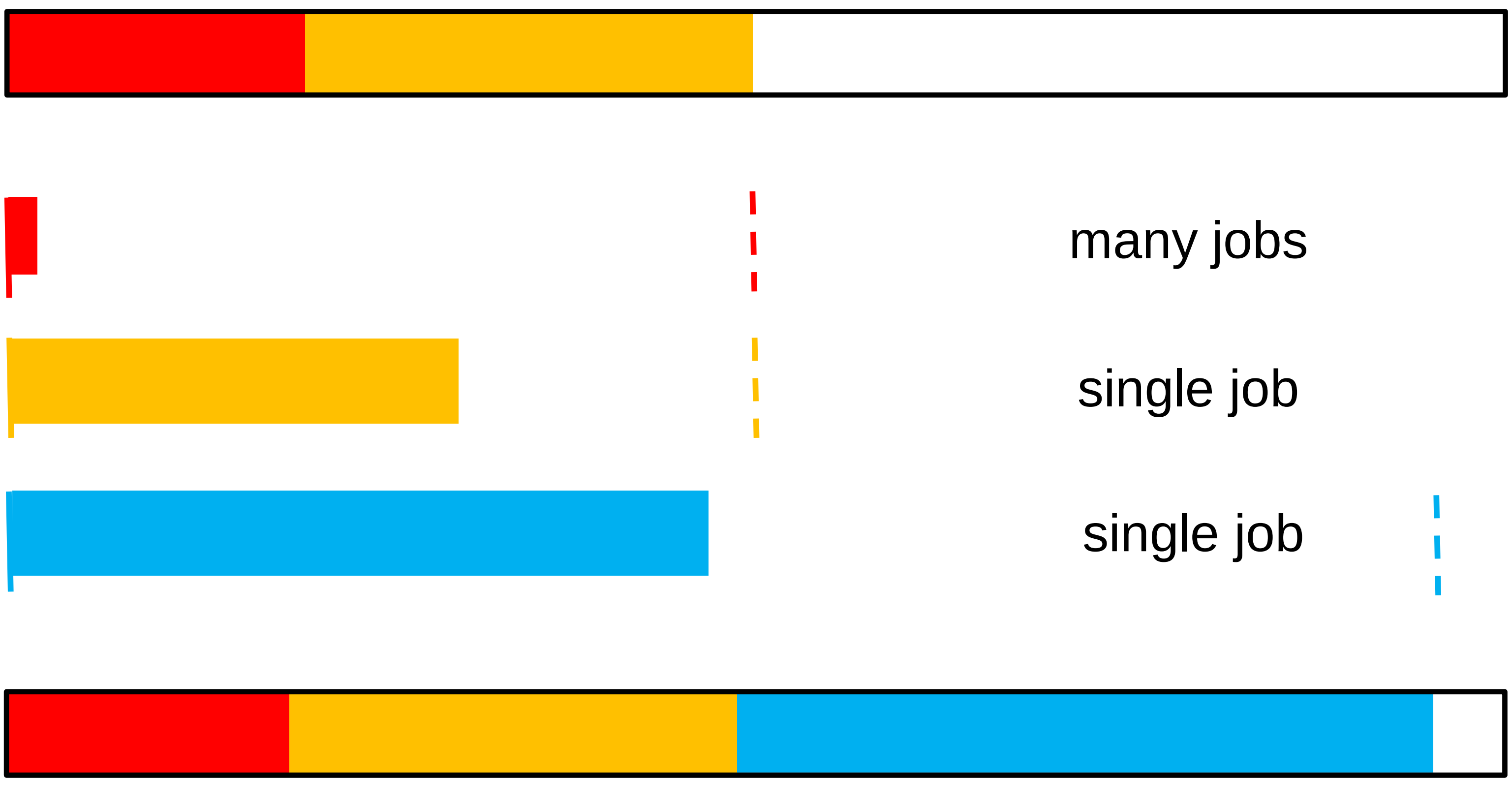
Single Machine Bound without Preemptions



Known Results with Preemptions

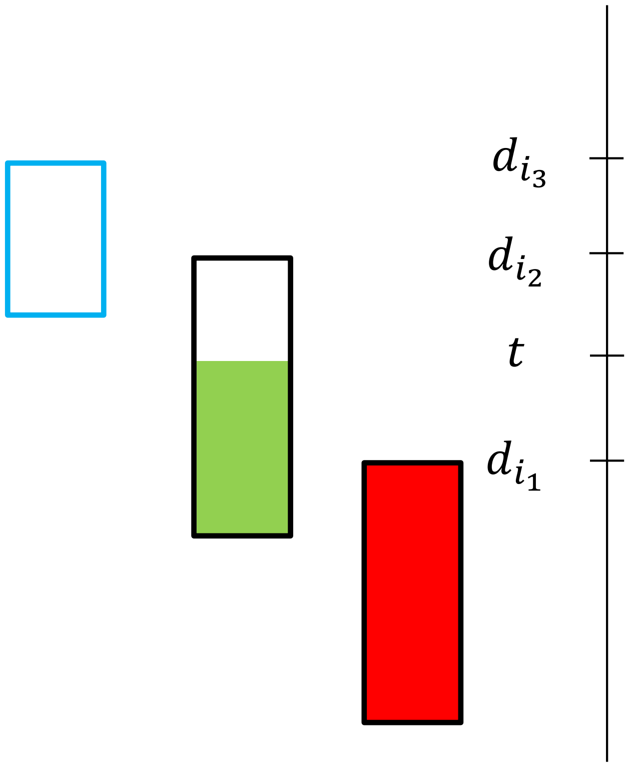
- Single machine results with preemption
  - Greedy is $1 + 1/\varepsilon$ competitive. The bound is tight (DasGupta and Palis 2000).
- Parallel identical machines with preemption and without migration
  - Transfer of the single machine algorithm to parallel identical machines (DasGupta and Palis 2000).
  - The competitive ratio is not correct for large slack values ε.
  - Lower bound $1 + 1/_{m \cdot \lceil 1+\varepsilon \rceil - 1}$ (DasGupta and Palis 2001).

Single Machine Bound with Preemptions



A Simple Check for a Legal Schedule

$V_{min}(t)$: minimum amount of processing volume in interval [0,t)



$$V_{min}(t) = \sum_{j \in J} \begin{cases} 0 & \text{for } d_j - p_j \geq t \\ t - d_j + p_j & \text{for } d_j > t > d_j - p_j \\ p_j & \text{for } t \geq d_j \end{cases}$$

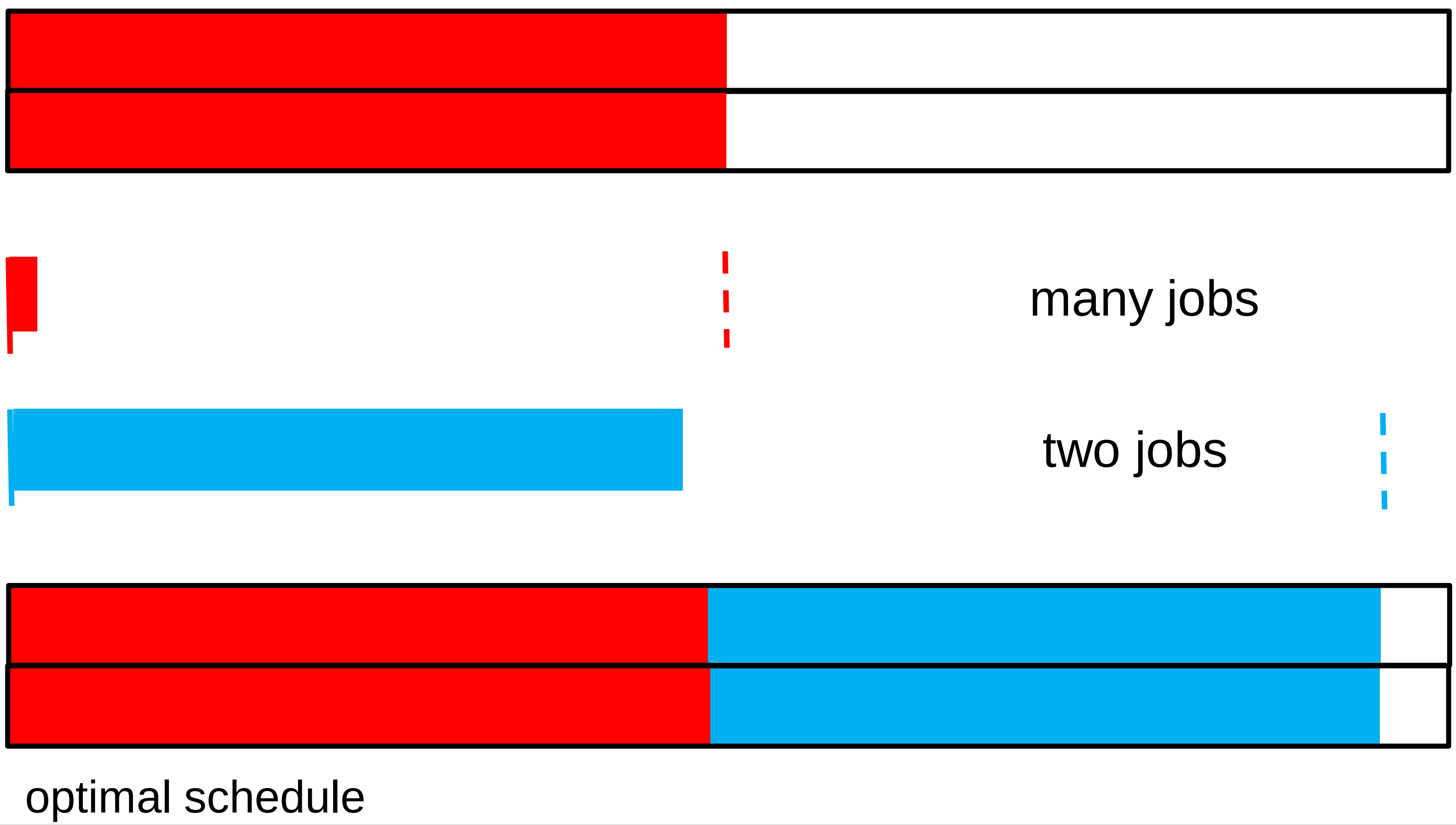
There is a legal schedule if and only if

$$V_{min}(t) \leq m \cdot t \text{ for all } t \geq 0$$

Greedy Acceptance Policy on *m* Parallel Identical Machines

- We accept a new job if there is a valid schedule that completes the new job and all previously accepted jobs in time.
- We use the V_{min} criterion to test whether there is a valid schedule.
- The worst case single machine scenario applies as well!

Greedy Acceptance Policy on 2 Parallel Identical Machines



Lazy Acceptance Policy

- We do not accept every job although it may allow a legal schedule.
- We exchange the criterion $V_{min}(t)$ by a criterion $V_{sim}(t) \leq V_{min}(t)$.
- We exchange our threshold $m \cdot t$ by $F(m, \varepsilon) \cdot t \leq m \cdot t$.

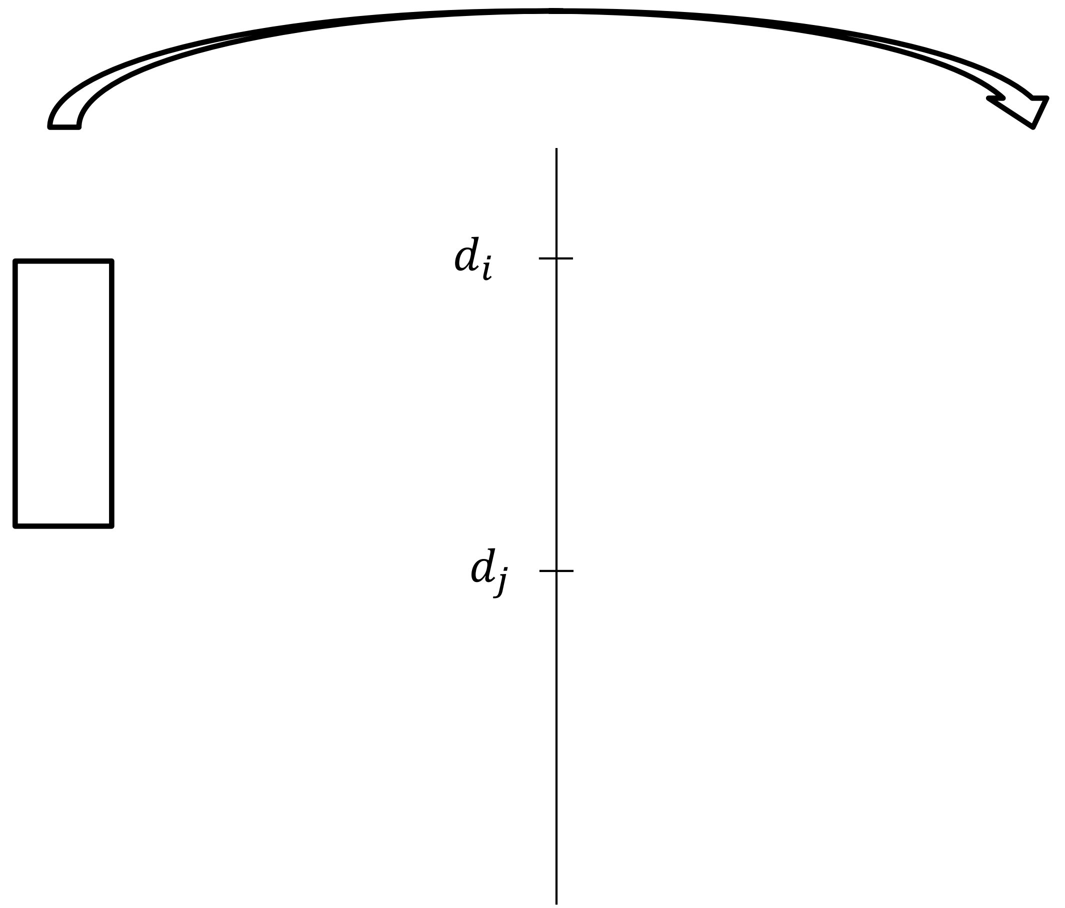
$$V_{sim}(t) = \sum_{j \in J} \begin{cases} 0 & \text{for } d_j > t \\ p_j & \text{for } d_j \leq t \end{cases}$$

$$f(m, \varepsilon) = \frac{\varepsilon}{1+\epsilon} \cdot \sum_{i=0}^{m-1} \left(\frac{1+\epsilon}{\epsilon}\right)^{\frac{i}{m}} \qquad F(m, \varepsilon) = \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}} \cdot f(m, \varepsilon)$$

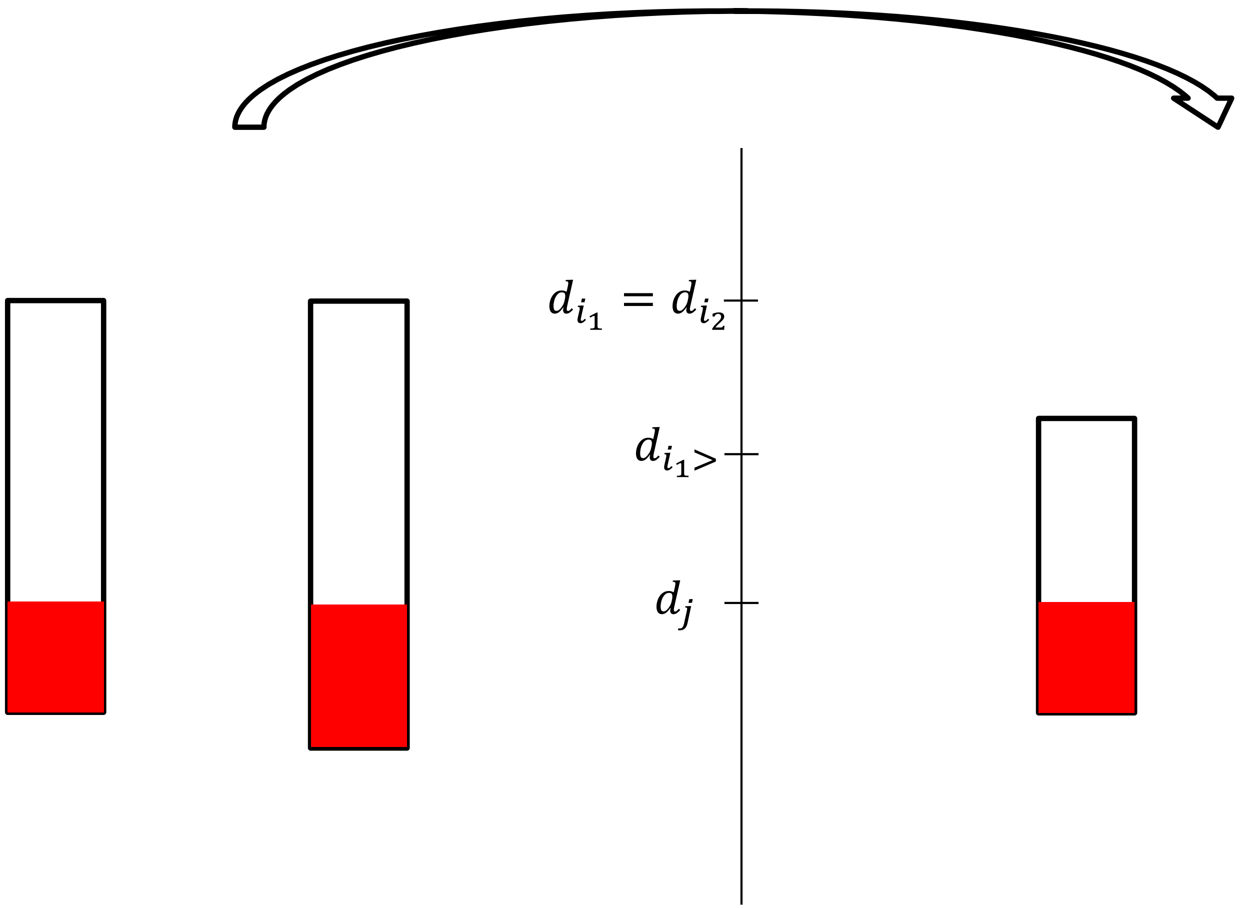
Proof of the Existence of a Legal Schedule (Key Lemma)

- We test whether $V_{min}(d_j) \leq d_j \cdot m$ always holds if the new criterion does not exceed the new threshold.
- We reduce the instance space that we must examine.
  - We apply transformations that cannot decrease $V_{min}(d_j)$ while they do not increase $V_{sim}(d_i)$ for any $d_i > d_j$.
- We analyze all job sequences of the reduced instance space.

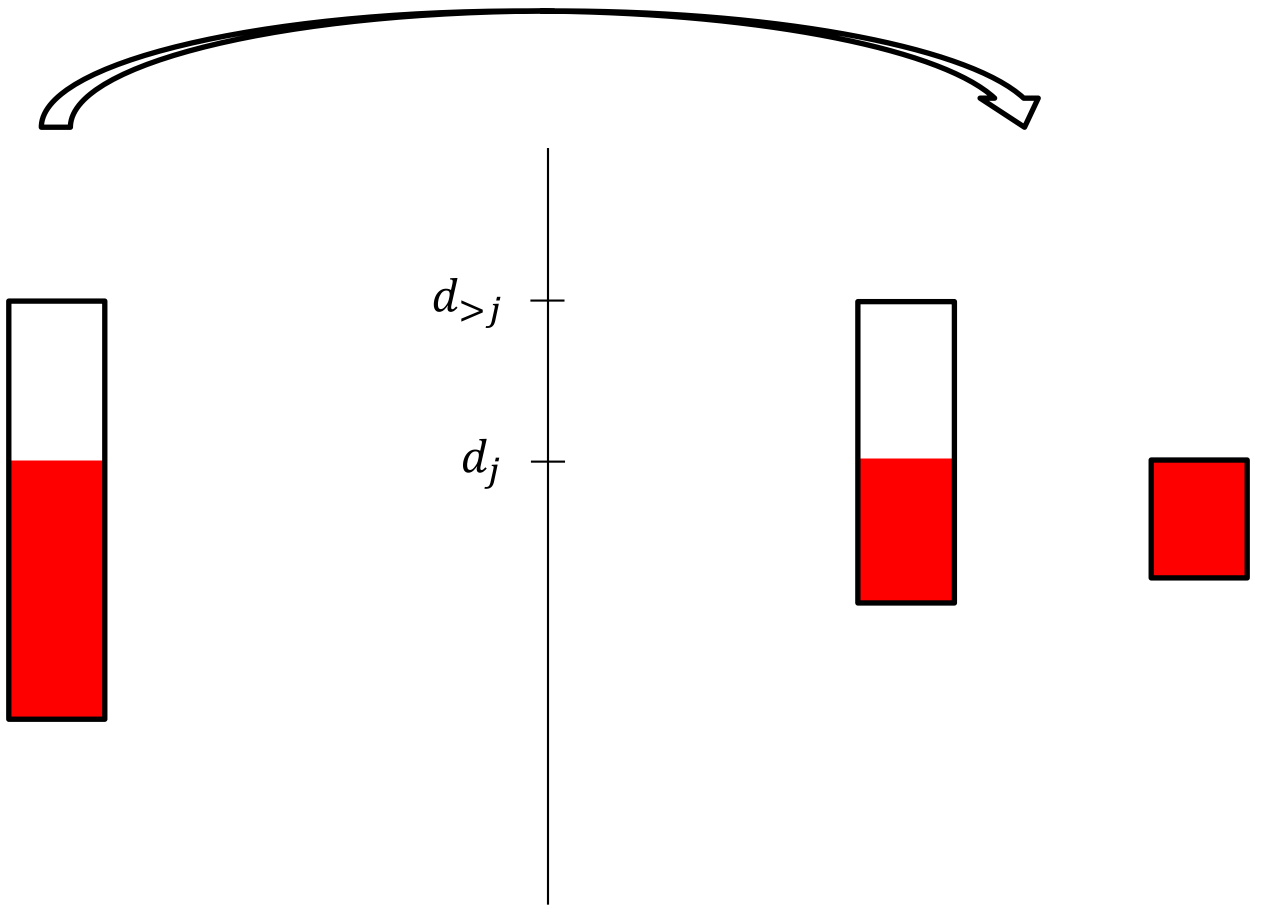
Job Removal Transformation



Spread Generation Transformation

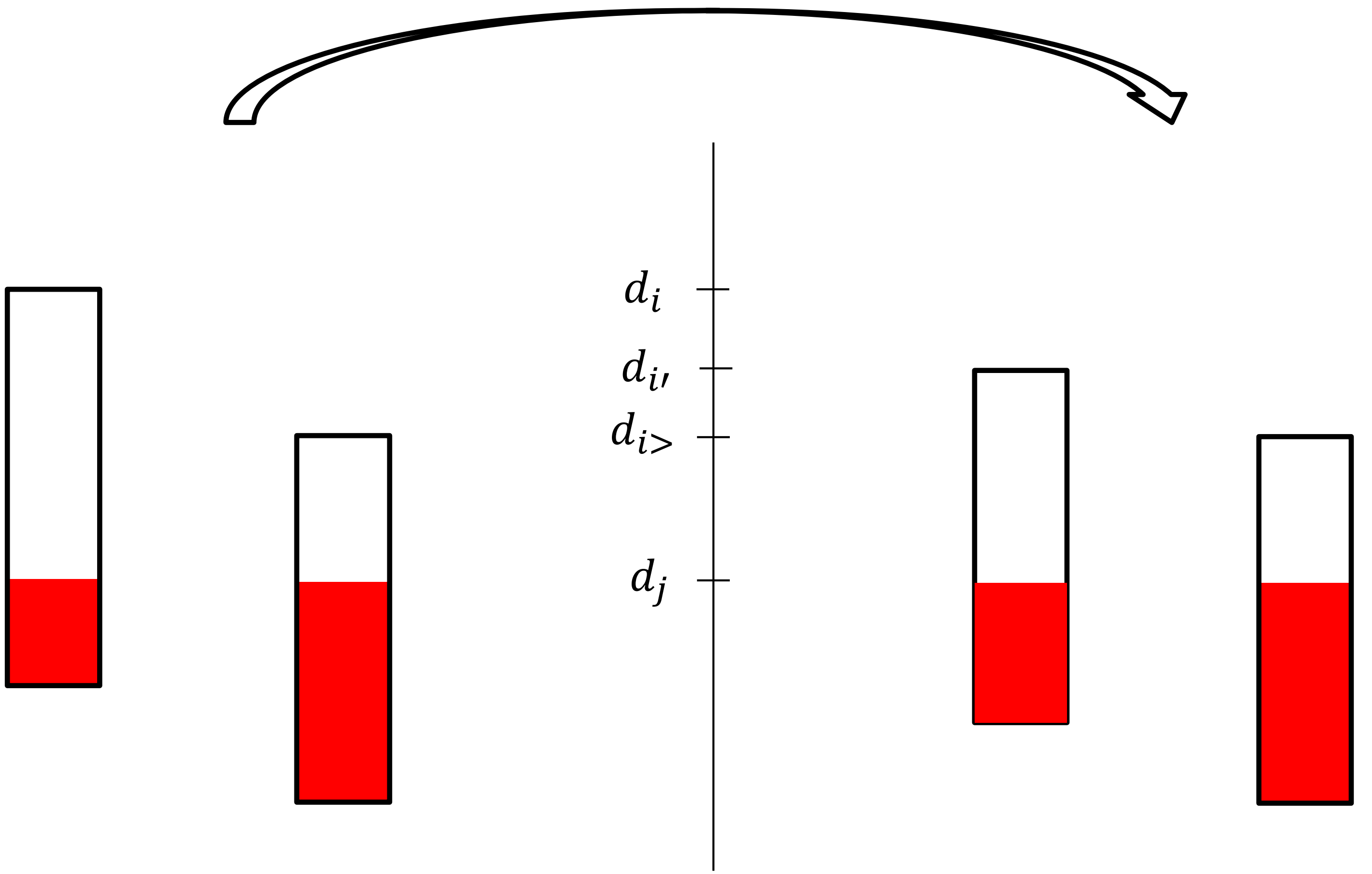


V_0-Transformation



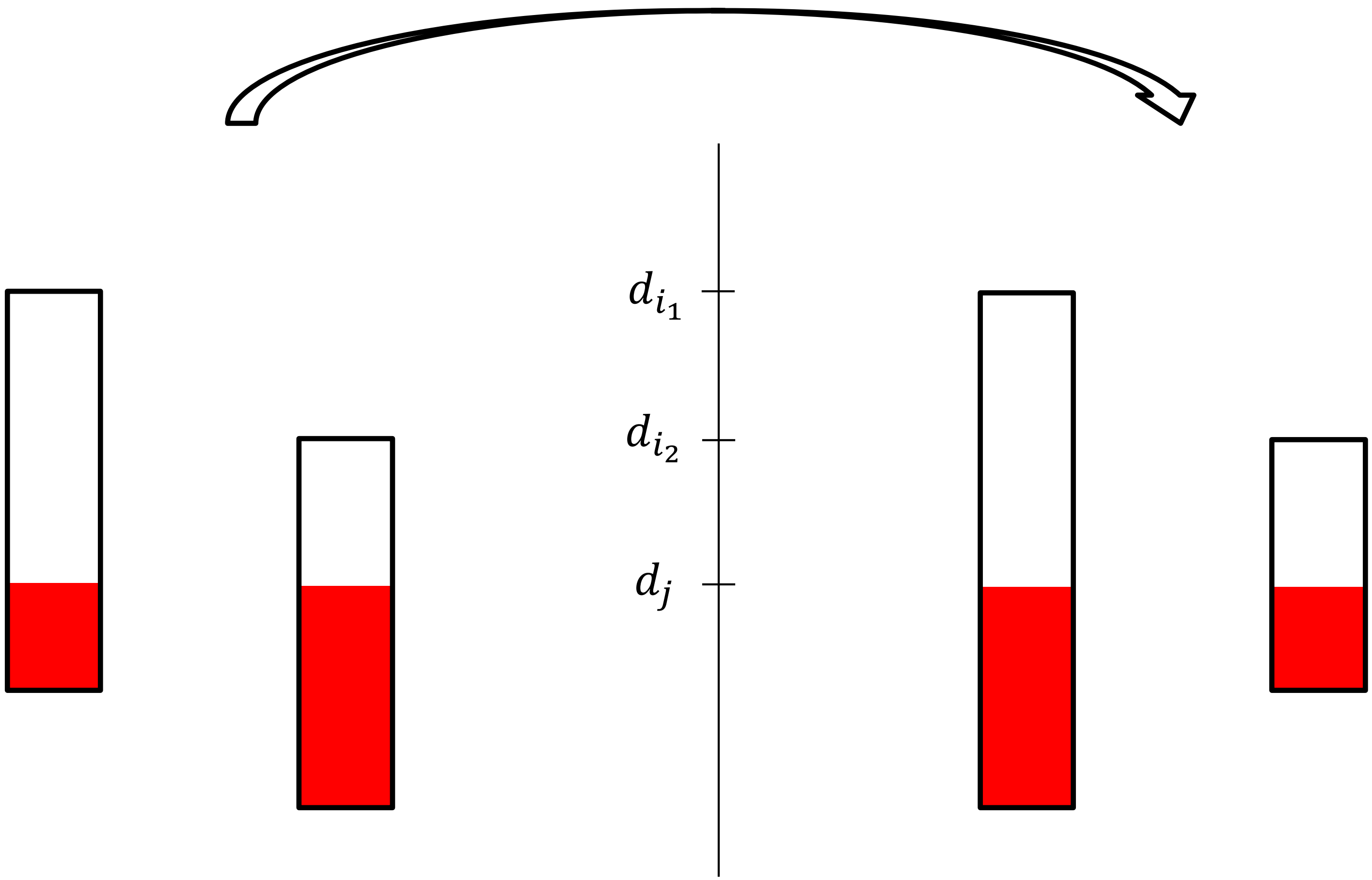
technische universität dortmund

V-Transformation



technische universität
dortmund

p-Transformation



Worst Case Sequence

1. $p_{>j} \leq \frac{d_{>j}}{1+\varepsilon}$
2. $p_i = \frac{d_i}{1+\varepsilon}$ for every deadline $d_i > d_{>j}$
3. $V_{sim}(d_i) = d_i \cdot F(m, \varepsilon)$for all $d_i \geq d_j$.
4. There are no jobs i with $d_i - p_j \geq d_j$.
5. There are no two jobs with the same deadline $d_i > d_j$.

Sequence of jobs with geometrically increasing deadlines and tight slack

$$d_i = d_{>i} \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}$$

Algorithm Limit test

- The algorithm uses $t_{min} = \mathrm{argmax}_{\tau}\{V_{min}(\tau)|_t = (\tau - t) \cdot f(m, \varepsilon)\}$.

- **For** each submitted job j **do**
  - use $r_j = t$ to determine t_{min};
  - **if** $d_j \geq t_{min}$ **then**
    - accept job j; update t_{min};
  - **else**
    - reject job j;
  - **end if**
- **end for**

Sequence Problem

- A sequence problem is an online problem in which all jobs are submitted at time 0.
  - But we must make our decision on any job before seeing the next job.
- In some online problems, worst cases occur in the corresponding sequence problem.
  - $1|\varepsilon, \text{online}, \text{pmtn}|\sum p_j \cdot (1 - U_j)$
- In other online problems, progression of time leads to more complexity and a larger competitive ratio.
  - $1|\varepsilon, \text{online}|\sum p_j \cdot (1 - U_j)$

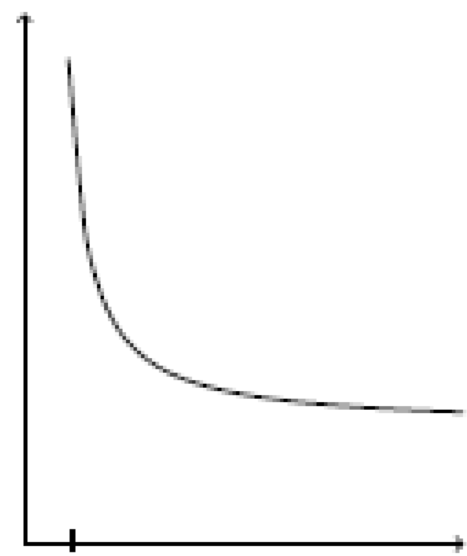
technische universität dortmund

Competitive Ratio with Preemption and Migration

$$P_m|\varepsilon, \text{sequence}, \text{pmtn}|\sum p_j \cdot \left(1 - U_j\right)$$

Competitive Ratio $\dfrac{m \cdot (1+\varepsilon)}{f(m,\varepsilon)}$

Lower Bound $\dfrac{\lfloor m \cdot (1+\varepsilon) \rfloor}{f(m,\varepsilon)}$

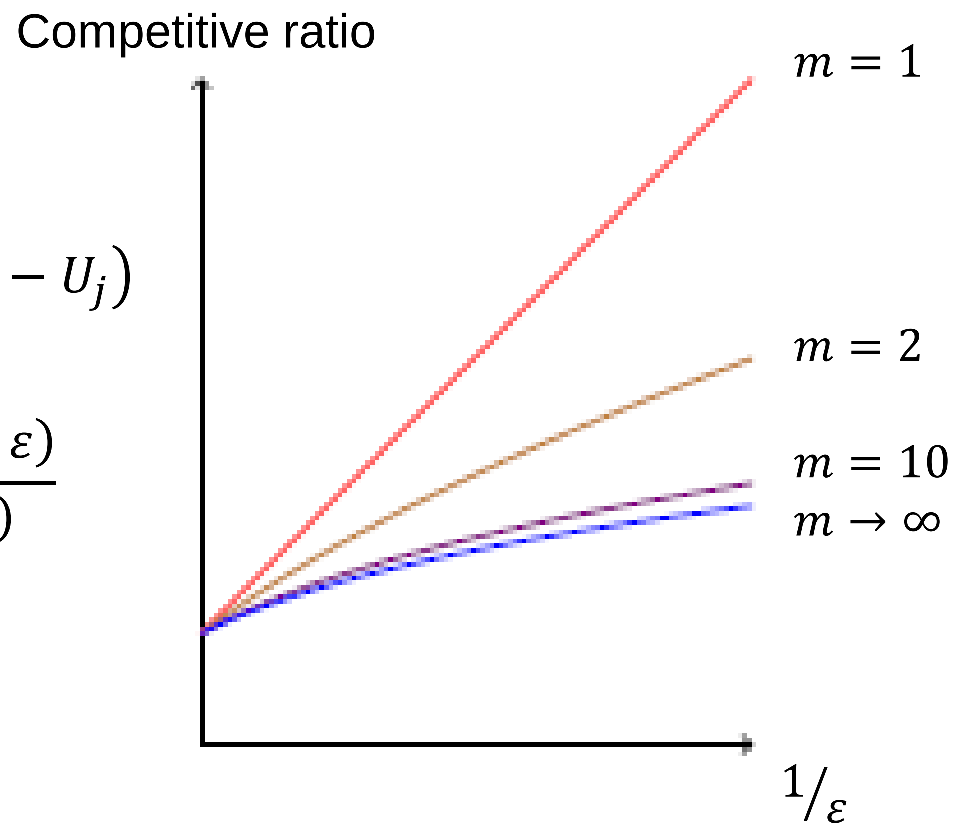


technische universität dortmund

Competitive Ratio and the Number of Machines

$$P_m|\varepsilon, \text{sequence}, \text{pmtn}|\sum p_j \cdot (1 - U_j)$$

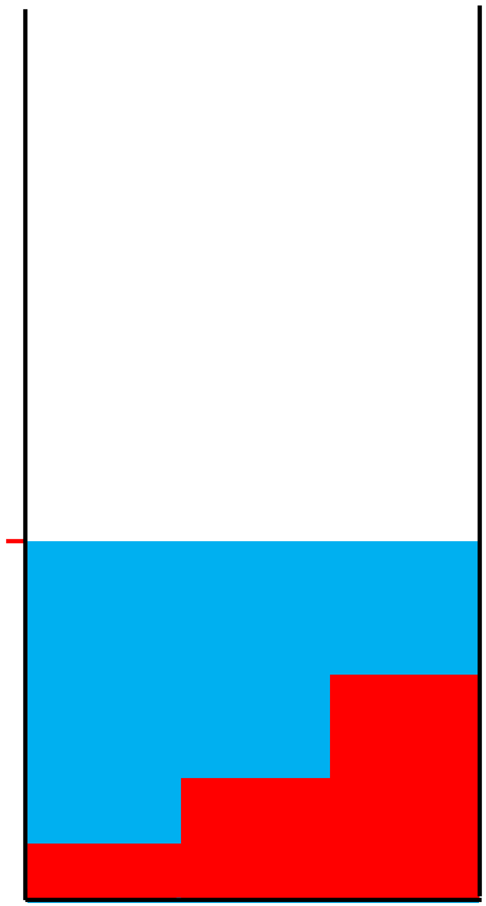
Competitive Ratio $\quad \dfrac{m \cdot (1+\varepsilon)}{f(m,\varepsilon)}$



technische universität
dortmund

Lower Bound Base Phase

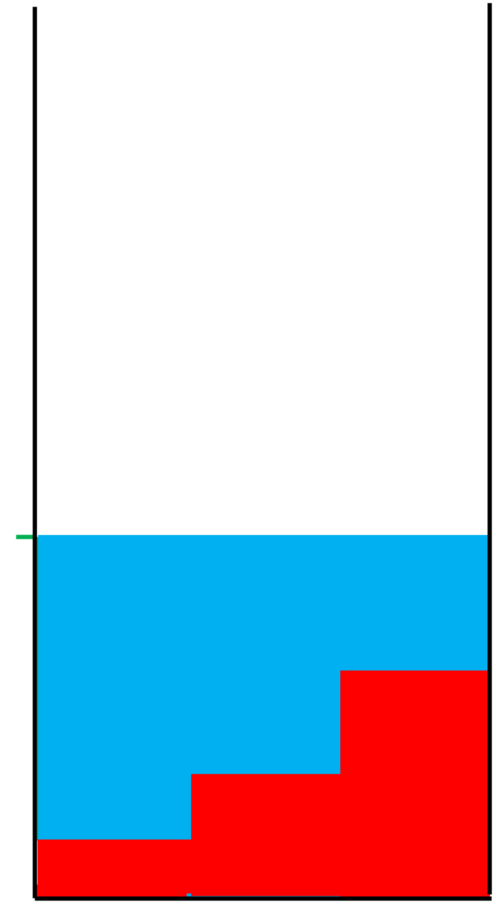
- Submission of many small jobs
- The adversary stops the submission if the planned area is covered with these jobs.
- If we do not accept enough small jobs then the adversary submits enough jobs to cover the whole area until the deadline.



technische universität
dortmund

Lower Bound Filling Phase

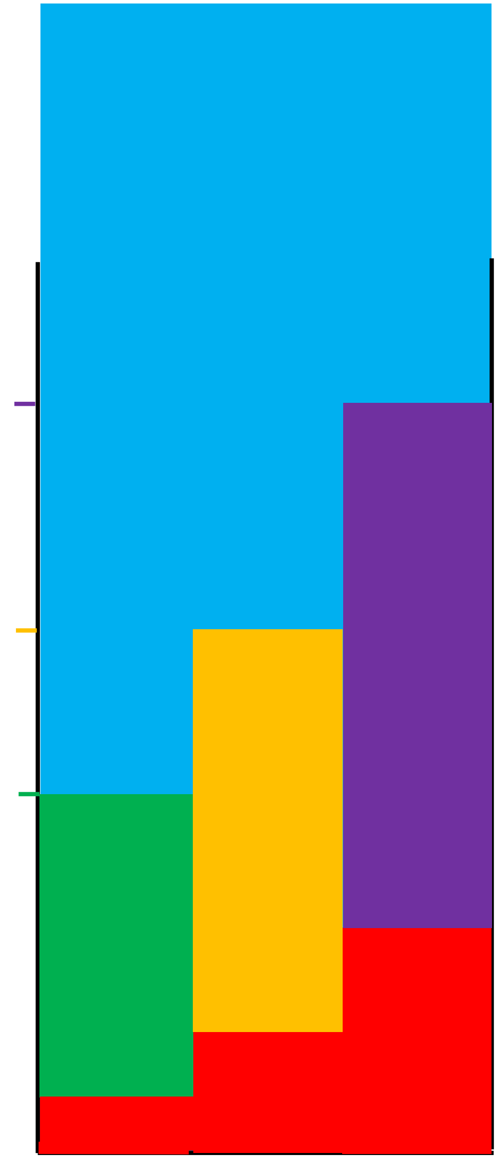
- Submission of long jobs with tight slack.
- The adversary stops the submission if we accept one job.
- If we do not accept a job then the adversary submits enough jobs to cover the area $\lfloor m \cdot (1+\varepsilon) \rfloor \cdot p_j$.



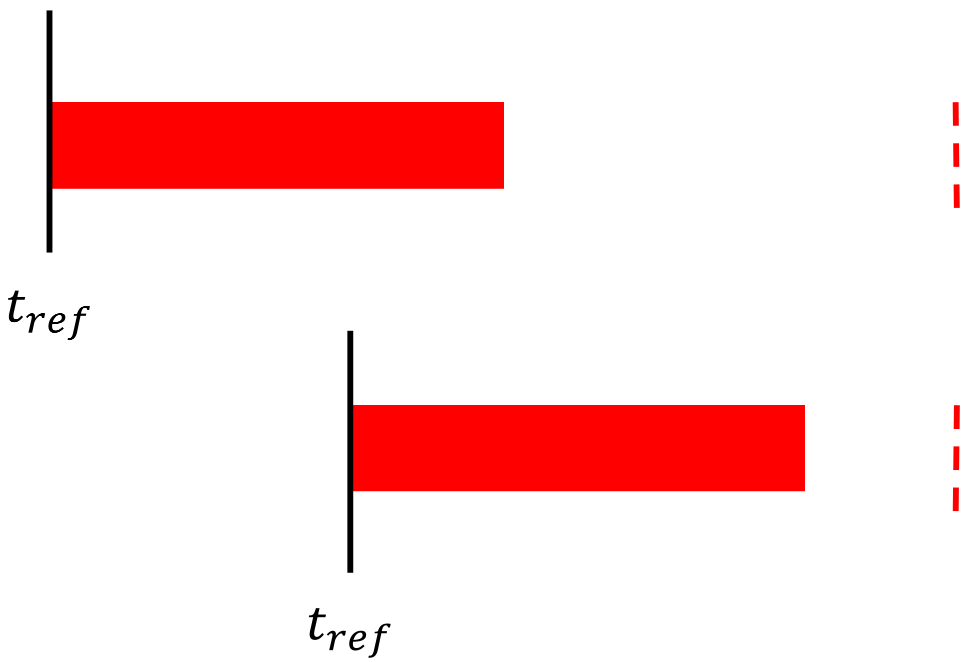
technische universität dortmund

Lower Bound Completion

- Repetition of this procedure with exponentially increasing processing times and deadlines.
- For the final job type, the adversary first exponentially increases the processing time and then reduces it by a very small amount δ. The adversary selects the deadline to generate a tight slack. We cannot accept any of these jobs.



Slack and the Progression of Time



tight slack at submission
for $\varepsilon = 1$

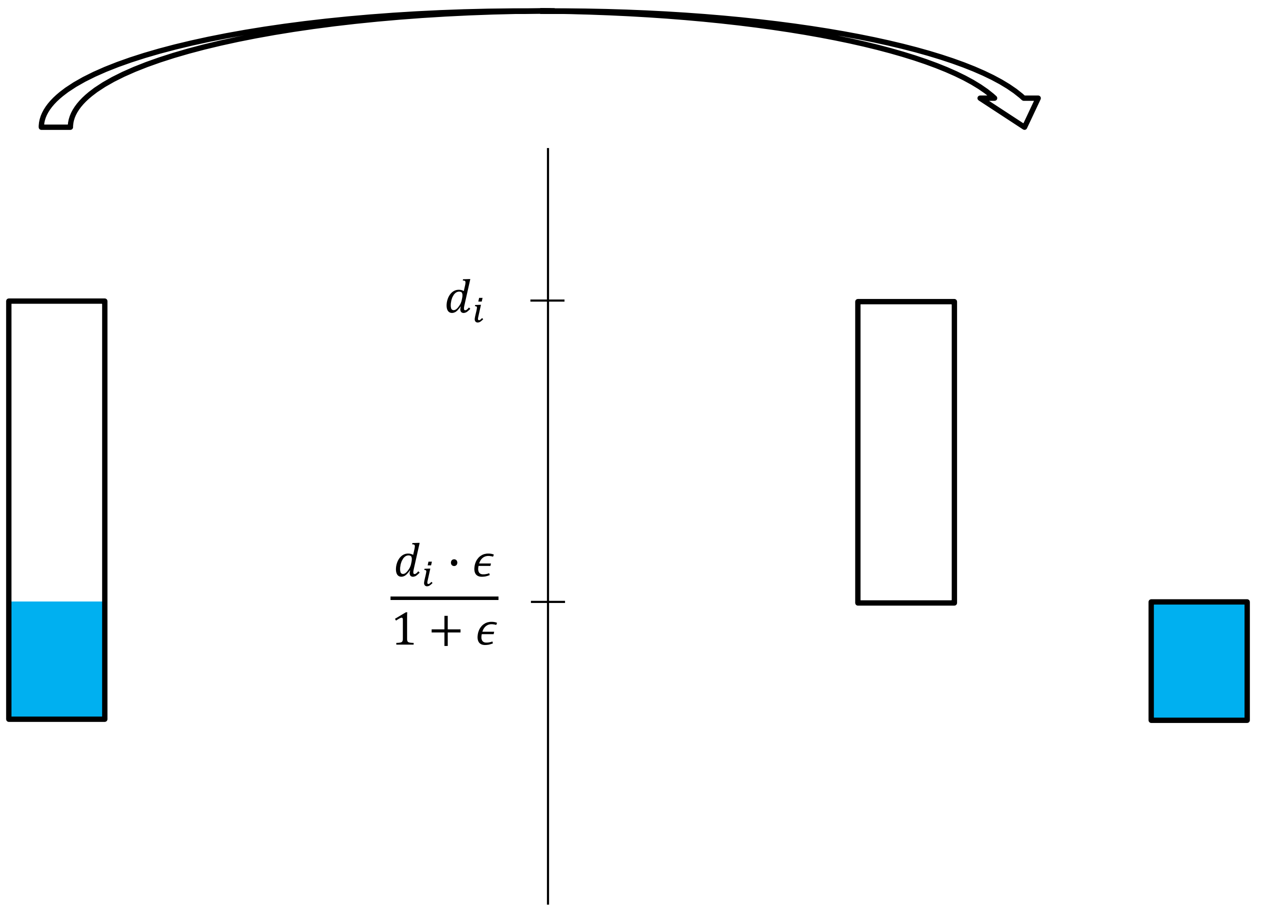
slack violation after
progression of time and
delayed execution

Modification of the Key Lemma

- We replace V_{sim} by a more complex criterion V_{acc}.
- We add a new transformation *large job splitting*.
- The lemma still holds.

$$V_{acc}(t) = \sum_{j \epsilon J} \begin{cases} 0 & \text{for } d_j - p_j \geq t \\ t - d_j + p_j & \text{for } d_j - {d_j}/{1+\varepsilon} \geq t > d_j - p_j \\ max\left\{p_j - {d_j}/{1+\varepsilon}, 0\right\} & \text{for } d_j > t > d_j - {d_j}/{1+\varepsilon} \\ p_j & \text{for } t \geq d_j \end{cases}$$

Large Job Splitting Transformation

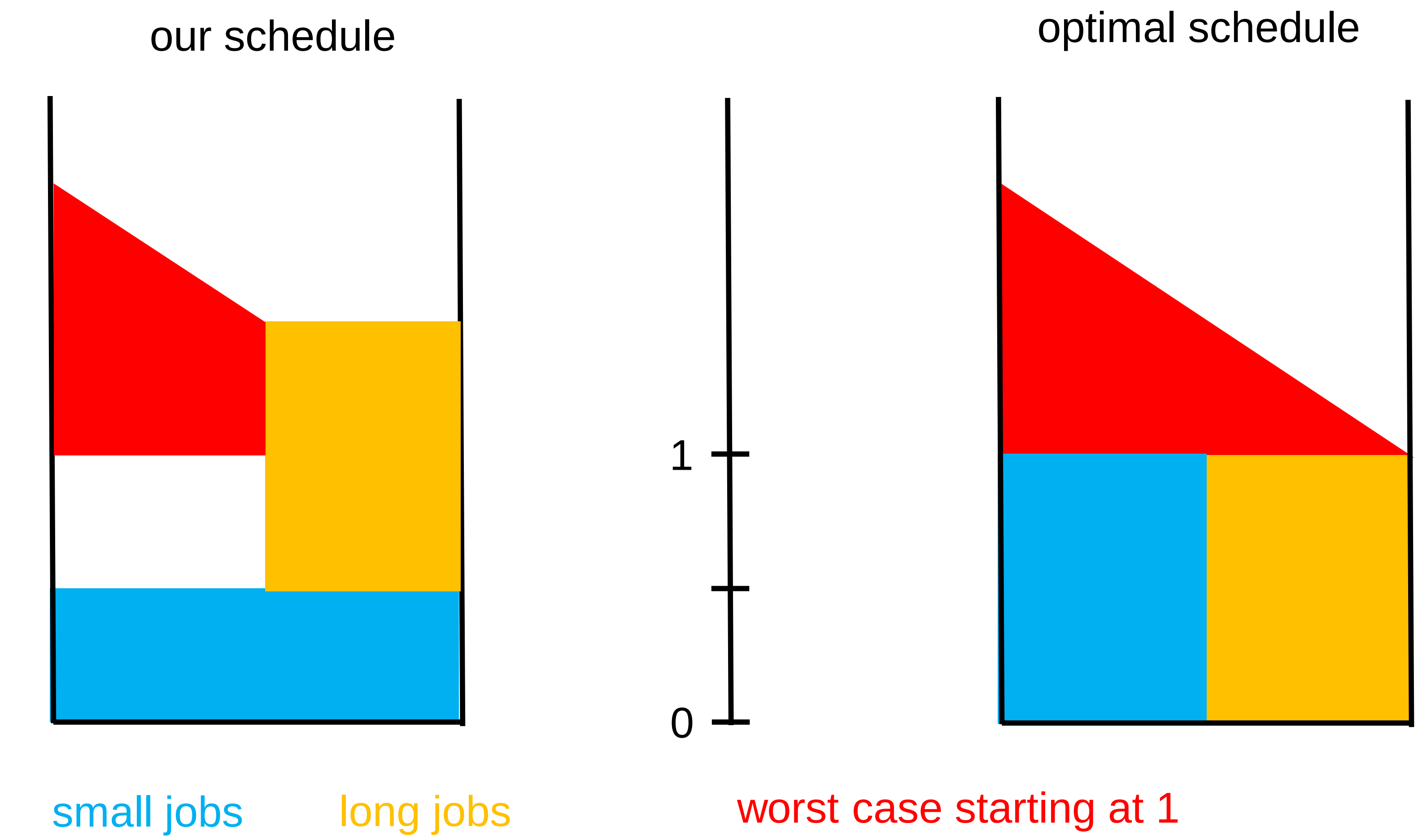


Impact of the Schedule

- The schedule has an impact on worst case sequence situations with a small competitive ratio (here large slack values).
- Many small jobs with deadline 1 and a total processing time $^{m}/_{2}$.
- $^{m}/_{2}$ jobs with processing time 1 and suitably large deadline.
- There are two basic strategies for allocation
  - Balancing
  - Concentration

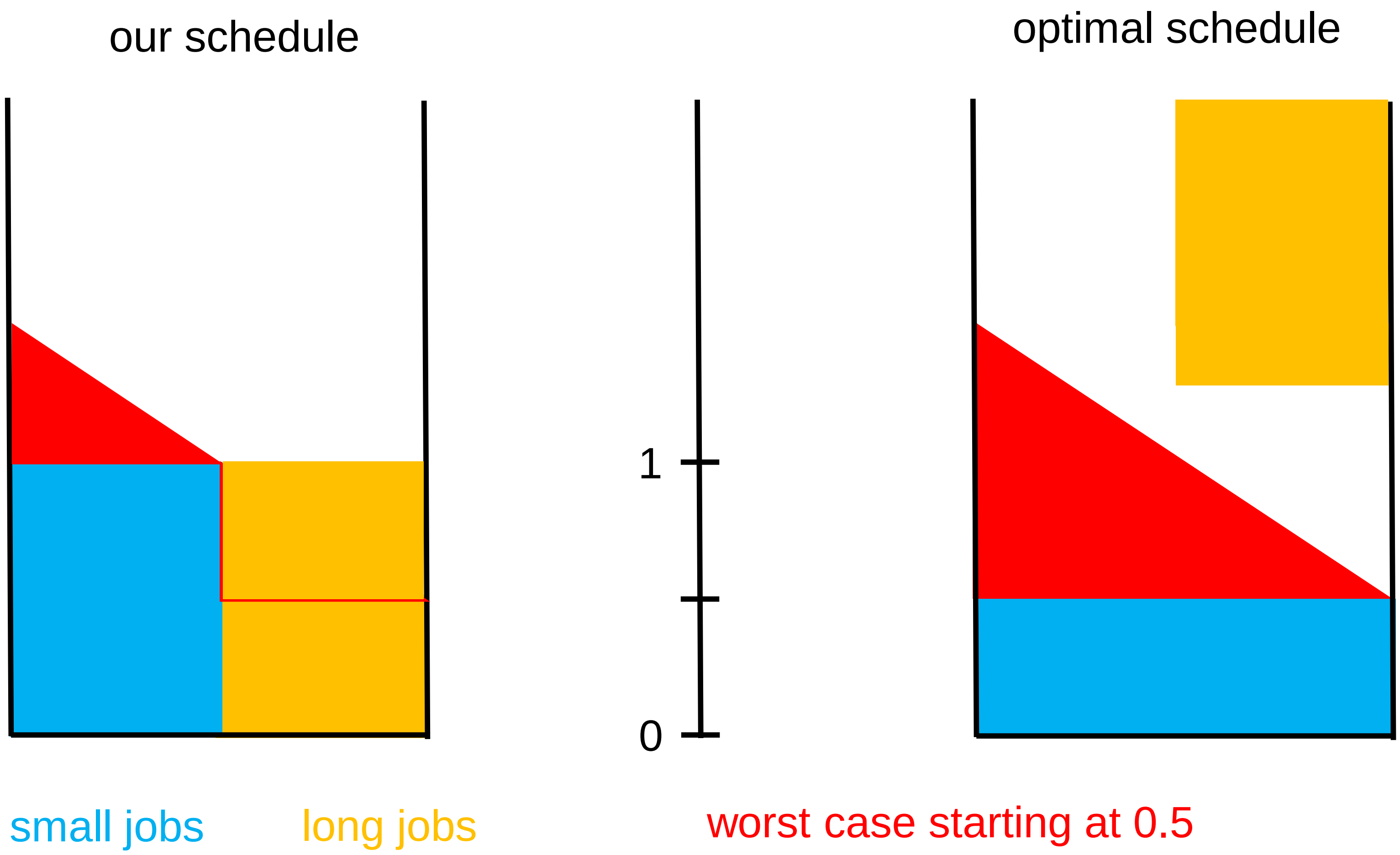
- Greedy scheduling with preemption has a worse competitive ratio on parallel identical machines than on a single machine for large slack values.

Allocation with Concentration Strategy



technische universität
dortmund

Allocation with Balancing Strategy



Competitive Ratio of the Greedy Acceptance Policy

- The greedy acceptance policy for the problem $P_m|\varepsilon, \text{pmtn}, \text{online}|\sum p_j \cdot (1 - U_j)$ has a competitive ratio of at least

$$\frac{4\varepsilon^2 + 14\epsilon + 2}{4\varepsilon^2 + 9}.$$

- This term is larger than $\frac{1+\varepsilon}{\varepsilon}$ for $\varepsilon > 7$.
- The result also holds for preemption without migration since the proof does not use migration. Therefore, the result corrects the claim by DasGupta and Palis (2000).

Influence of Progression of Time on the Threshold

- Our algorithm guarantees the validity of the threshold for all times in the sequence problem: $V_{sim}(t) \leq F(m, \varepsilon) \cdot t$
- After progression of time, this condition may not hold anymore.
  - The condition for our key lemma is not true anymore!
- We can prove that the algorithm still works when we use allocation with concentration strategy.
- Competitive ratio $max\left\{\frac{m \cdot (1+\varepsilon)}{f(m,\varepsilon)}, \frac{m \cdot (1+\varepsilon)}{4f(m,\varepsilon)} + 1\right\}$

Impact of allocation with concentration strategy for large values of ε.

Restrictions in Practice

- Preemption with migration for resource management
  - We use migration in case of failures. Can we use it for resource management?
- Heterogeneity of resources
  - Most large computer systems are not homogeneous.
- Multiple resources
  - The analysis only considers a single resource. How about computation and bandwidth and storage?
- Priority of some jobs
  - Can we handle jobs with different priorities?
- Type of jobs
  - How can we handle interactive or parallel jobs?