An $O(\log m)$-Competitive Algorithm for Online Machine Minimization

Nicole Megow
with Lin Chen and Kevin Schewior

Technische Universität München

Workshop on Approximation Algorithms and Parameterized Complexity
Banff, December 2015
The Problem: Definition

- **Input**: set of preemptable jobs \( J = \{1, \ldots, n\} \) where each job \( j \) is
The Problem: Definition

- **Input**: set of preemptable jobs $J = \{1, \ldots, n\}$ where each job $j$ is

  - release date $r_j$,
  - deadline $d_j$

  Task: Find a feasible schedule on a min. number of machines.

  - Each job $j$ is processing for $p_j$ time units within $[r_j, d_j]$.
  - At any time, any job runs on at most one machine.
The Problem: Definition

- **Input**: set of preemptable jobs $J = \{1, \ldots, n\}$ where each job $j$ is

  - \text{release date } r_j, \quad \text{deadline } d_j
  
  \begin{center}
  \begin{tikzpicture}
  \draw[->] (0,0) -- (5,0) node[below] {time};
  \draw (0,0) -- (1,0); \node at (0.5,0.2) {0};
  \draw (1,0) -- (2,0); \node at (1.5,0.2) {1};
  \draw (2,0) -- (5,0);
  \draw (2,0.2) -- (2,0.6);
  \draw (3,0) -- (5,0) node[midway,above] {processing time } node[midway,below] {p_j};
  \end{tikzpicture}
  \end{center}
The Problem: Definition

- **Input**: set of preemptable jobs $J = \{1, \ldots, n\}$ where each job $j$ is
  - release date $r_j$,
  - deadline $d_j$,
  - processing time $p_j$.

- **Task**: Find a feasible schedule on a min. number of machines.
The Problem: Definition

- **Input**: set of preemptable jobs \( J = \{1, \ldots, n\} \) where each job \( j \) is
  
  
  
  release date \( r_j \), deadline \( d_j \)
  
  
  
  processing time \( p_j \)

- **Task**: Find a feasible schedule on a min. number of machines.
  - Each job \( j \) is processing for \( p_j \) time units within \([r_j, d_j]\).
The Problem: Definition

- **Input**: set of preemptable jobs $J = \{1, \ldots, n\}$ where each job $j$ is

  release date $r_j$, deadline $d_j$

  0 1 ---------------------------------------- time

  processing time $p_j$

- **Task**: Find a **feasible** schedule on a **min. number of machines**.
  - Each job $j$ is processing for $p_j$ time units within $[r_j, d_j]$.
  - At any time, any job runs on at most one machine.
**The Problem: Definition**

- **Input**: set of preemptable jobs $J = \{1, \ldots, n\}$ where each job $j$ is

  - release date $r_j$, 
  - deadline $d_j$

![Diagram of release date and deadline](image)

- **Task**: Find a feasible schedule on a min. number of machines.
  - Each job $j$ is processing for $p_j$ time units within $[r_j, d_j]$.
  - At any time, any job runs on at most one machine.

![Diagram of schedule](image)
The Problem: Definition

- **Input**: set of preemptable jobs $J = \{1, \ldots, n\}$ where each job $j$ is
  - release date $r_j$,
  - deadline $d_j$,
  - processing time $p_j$.

- **Task**: Find a feasible schedule on a min. number of machines.
  - Each job $j$ is processing for $p_j$ time units within $[r_j, d_j]$.
  - At any time, any job runs on at most one machine.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn '74
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn ’74

**Online**: A job becomes known only at its release date.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn ’74

**Online**: A job becomes known only at its release date.

Slack versus processing time?
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn '74

**Online**: A job becomes *known only at its release date*.

Slack versus processing time?
The Problem: Offline vs. Online

**Offline:** Optimally solvable in polynomial time (LP or max flow). Horn ’74

**Online:** A job becomes known only at its release date.

Slack versus processing time?

**Case 1:**
The blue job is scheduled to some extent in [0, 1], i.e., some green job unfinished by 1.

Online:
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn ’74

**Online**: A job becomes known only at its release date.

Slack versus processing time?

Case 1:

The blue job is scheduled to some extent in [0, 1], i.e., some green job unfinished by 1.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn '74

**Online**: A job becomes known only at its release date.

Slack versus processing time?

Case 1:
The blue job is scheduled to some extent in [0, 1], i.e., some green job unfinished by 1.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn '74

**Online**: A job becomes known only at its release date.

Slack versus processing time?

Case 1:
The blue job is scheduled to some extent in [0, 1], i.e., some green job unfinished by 1.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn '74

**Online**: A job becomes *known only at its release date*.

Slack versus processing time?

**Case 2**: The blue job is *not* scheduled in [0, 1], i.e., does not finish by 2.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn ’74

**Online**: A job becomes known only at its release date.

---

Slack versus processing time?

**Case 2**:

The blue job is *not* scheduled in $[0, 1]$, i.e., does not finish by 2.

[Diagram showing time intervals and job schedules]
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn ’74

**Online**: A job becomes known only at its release date.

Slack versus processing time?

---

**Case 2:**

The blue job is not scheduled in [0, 1], i.e., does not finish by 2.
The Problem: Offline vs. Online

**Offline**: Optimally solvable in polynomial time (LP or max flow). Horn '74

**Online**: A job becomes known only at its release date.

---

Slack versus processing time?

Case 2:

The blue job is not scheduled in [0, 1], i.e., does not finish by 2.
The Problem: Online

**Bad news:** No (preemptive) online algorithm can guarantee to find always a feasible schedule on the minimum number of machines.

Dertousoz and Mok (TSE 1989)
The Problem: Online

**Bad news:** No (preemptive) online algorithm can guarantee to find always a feasible schedule on the minimum number of machines.

Dertousoz and Mok (TSE 1989)

**Competitive analysis:** An online algorithm ALG is *c*-competitive if

\[ ALG(I) \leq c \cdot m(I), \]

for all instances \( I \) with a feasible offline schedule on \( m(I) \) machines.
Phillips et al. (STOC 1996)

- Best known algorithm (LLF) is $\mathcal{O}(\log \frac{p_{\text{max}}}{p_{\text{min}}})$-competitive.
- No deterministic algorithm is $(5/4 - \varepsilon)$-competitive.
Phillips et al. (STOC 1996)

- Best known algorithm (LLF) is $O(\log \frac{p_{\text{max}}}{p_{\text{min}}})$-competitive.
- No deterministic algorithm is $(\frac{5}{4} - \varepsilon)$-competitive.
- Open if there is an $f(m)$-competitive algorithm for any fct. $f$. 
Previous Results

Phillips et al. (STOC 1996)

- Best known algorithm (LLF) is $O(\log \frac{p_{\max}}{p_{\min}})$-competitive.
- No deterministic algorithm is $(\frac{5}{4} - \varepsilon)$-competitive.

- Open if there is an $f(m)$-competitive algorithm for any fct. $f$.
- No better result even for $m = 2$. 
Our Contribution

Theorem

There is an online algorithm with competitive ratio $O(\log m)$. 
Our Contribution

**Theorem**

There is an online algorithm with competitive ratio $O(\log m)$.

**Key ingredients**

1. New algorithm carefully balancing the delay of tight jobs.
2. New lower bound relating number and laxity of critical jobs.
Our Contribution

**Theorem**

There is an online algorithm with competitive ratio $O(\log m)$.

**Key ingredients**

1. New algorithm carefully balancing the delay of tight jobs.
2. New lower bound relating number and laxity of critical jobs.

At the loss of a factor 4, we may assume $m$ is known.

$\rightarrow$ Guess-and-Double
Earliest Deadline First

**Algorithm EDF\(_{m'}\):** At any time schedule \( m' \) available jobs with minimum deadline and preempt other jobs if necessary.
Earliest Deadline First

Algorithm $\text{EDF}_{m'}$: At any time schedule $m'$ available jobs with minimum deadline and preempt other jobs if necessary.

**Theorem** [Phillips et al., STOC 1997]

There are instances (for any $m \geq 2$) on which $\text{EDF}_{n-1}$ fails.
Algorithm $EDF_{m'}$: At any time schedule $m'$ available jobs with minimum deadline and preempt other jobs if necessary.

**Theorem** [Phillips et al., STOC 1997]

There are instances (for any $m \geq 2$) on which $EDF_{n-1}$ fails.
Earliest Deadline First

Algorithm $\text{EDF}_{m'}$: At any time schedule $m'$ available jobs with minimum deadline and preempt other jobs if necessary.

**Theorem** [Phillips et al., STOC 1997]

There are instances (for any $m \geq 2$) on which $\text{EDF}_{n-1}$ fails.

Nicole Megow
Earliest Deadline First

Algorithm EDF

At any time schedule $m'$ available jobs with minimum deadline and preempt other jobs if necessary.

**Theorem** [Phillips et al., STOC 1997]

There are instances (for any $m \geq 2$) on which EDF$_{n-1}$ fails.

**Def.** Let $\alpha < 1$. Job $j$ is $\alpha$-tight if $p_j > \alpha(d_j - r_j)$ and $\alpha$-loose othw.
Algorithm $EDF_{m'}$: At any time schedule $m'$ available jobs with minimum deadline and preempt other jobs if necessary.

**Theorem** [Phillips et al., STOC 1997]

There are instances (for any $m \geq 2$) on which $EDF_{n-1}$ fails.

**Def.** Let $\alpha < 1$. Job $j$ is $\alpha$-tight if $p_j > \alpha(d_j - r_j)$ and $\alpha$-loose othw.

**Theorem** [Chen, M., Schewior (2015)]

If every job is $\alpha$-loose, then $EDF_\ast$ is $\frac{1}{(1-\alpha)^2}$-competitive.
Least Laxity First (LLF)

Laxity of a job

\[ l_j(t) \]

Algorithm Least Laxity First: At any time schedule the jobs with minimum laxity and preempt other jobs if necessary.

Theorem [Phillips et al., STOC 1997]
LLF may fail on \( f(m) \) machines, for any \( f \).

Theorem [Chen, M., Schewior 2015]
LLF may fail on \( f(m) \) machines, for any \( f \), even if jobs are \( \alpha \)-tight.
Least Laxity First (LLF)

Laxity of a job

\[ \ell_j(t) \]

Algorithm Least Laxity First: At any time schedule the jobs with minimum laxity and preempt other jobs if necessary.

Theorem [Phillips et al., STOC 1997] LLF may fail on \( f(m) \) machines, for any \( f \).

Theorem [Chen, M., Schewior 2015] LLF may fail on \( f(m) \) machines, for any \( f \), even if jobs are \( \alpha \)-tight.
Least Laxity First (LLF)

Laxity of a job

Laxity \( \ell_j(t) = d_j - t - p_j(t) \)
Least Laxity First (LLF)

Laxity of a job

\[
l_j(t) = d_j - t - p_j(t)
\]

Algorithm Least Laxity First: At any time schedule the jobs with minimum laxity and preempt other jobs if necessary.
Least Laxity First (LLF)

Laxity of a job

\[ \text{laxity } \ell_j(t) = d_j - t - p_j(t) \]

Algorithm Least Laxity First: At any time schedule the jobs with minimum laxity and preempt other jobs if necessary.

Theorem [Phillips et al., STOC 1997]

LLF may fail on \( f(m) \) machines, for any \( f \).
Least Laxity First (LLF)

Laxity of a job

\[ \ell_j(t) = d_j - t - p_j(t) \]

**Algorithm Least Laxity First:** At any time schedule the jobs with minimum laxity and preempt other jobs if necessary.

**Theorem** [Phillips et al., STOC 1997]

LLF may fail on \( f(m) \) machines, for any \( f \).

**Theorem** [Chen, M., Schewior 2015]

LLF may fail on \( f(m) \) machines, for any \( f \), even if jobs are \( \alpha \)-tight.
A New Lower Bound – Tight Jobs Only

Load-based lower bounds.
Load-based lower bounds.

Definition: Let $(\mu, \beta)$-critical pair $(G, T)$

1. Each time $t \in T$ is covered by $\geq \mu$ distinct jobs in $G$.
2. $|T \cap I(j)| \geq \beta \cdot \ell_j$, for any $j \in G$.

Theorem. If there is a $(\mu, \beta)$-critical pair then $m = \Omega(\mu \log 1/\beta)$.

Given $m$: If there is a $(\mu, \beta)$-critical pair then $\mu = O(m \cdot \log 1/\beta)$. 
A New Lower Bound – Tight Jobs Only

Load-based lower bounds.

Definition: 

(i) each time $t \in T$ is covered by $\geq \mu$ distinct jobs in $G$

(ii) $|T \cap I(j)| \geq \beta \cdot \ell_j$, for any $j \in G$

Theorem. If there is a $(\mu, \beta)$-critical pair then $m = \Omega(\mu \log \frac{1}{\beta})$.

Given $m$: If there is a $(\mu, \beta)$-critical pair then $\mu = O(m \cdot \log \frac{1}{\beta})$. 
A New Lower Bound – Tight Jobs Only

Relate laxity and number of intersecting intervals.

Definition: \((\mu, \beta)-\text{critical pair} (G, T)\)

(i) each time \(t \in T\) is covered by \(\geq \mu\) distinct jobs in \(G\)

(ii) \(|T \cap I(j)| \geq \beta \cdot \ell_j\), for any \(j \in G\)

Theorem. If there is a \((\mu, \beta)-\)critical pair then \(m = \Omega(\mu \log \frac{1}{\beta})\).

Given \(m\): If there is a \((\mu, \beta)-\)critical pair then \(\mu = O(m \cdot \log \frac{1}{\beta})\).
A New Lower Bound – Tight Jobs Only

Relate laxity and number of intersecting intervals.

Definition: \((\mu, \beta)\)-critical pair \((G, T)\)

(i) each time \(t \in T\) is covered by \(\geq \mu\) distinct jobs in \(G\)

(ii) \(|T \cap I(j)| \geq \beta \cdot \ell_j\), for any \(j \in G\)
Relate laxity and number of intersecting intervals.

Definition: \((\mu, \beta)\)-critical pair \((G, T)\)

(i) each time \(t \in T\) is covered by \(\geq \mu\) distinct jobs in \(G\)

(ii) \(|T \cap I(j)| \geq \beta \cdot \ell_j\), for any \(j \in G\)

Theorem. If there is a \((\mu, \beta)\)-critical pair then \(m = \Omega(\frac{\mu}{\log 1/\beta})\).
A New Lower Bound – Tight Jobs Only

Relate laxity and number of intersecting intervals.

Definition: \((\mu, \beta)\)-critical pair \((G, T)\)

(i) each time \(t \in T\) is covered by \(\geq \mu\) distinct jobs in \(G\)
(ii) \(|T \cap I(j)| \geq \beta \cdot \ell_j\), for any \(j \in G\)

Theorem. If there is a \((\mu, \beta)\)-critical pair then \(m = \Omega(\frac{\mu}{\log 1/\beta})\).

Given \(m\): If there is a \((\mu, \beta)\)-critical pair then \(\mu = O(m \cdot \log \frac{1}{\beta})\).
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity (= budget).
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity (\(=\) budget).

![Diagram showing open machines and their allocation of budget]

Open $m'$ machines. Charge the delay of a job to its laxity (\(=\) budget).
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity ($= \text{budget}$).

Use $i$-th budget (originally $\ell_j/(m' + 1)$) when $i - 1$ other jobs are active.
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity (\(=\) budget).

Use $i$-th budget (originally $\ell_j/(m' + 1)$) when $i - 1$ other jobs are active.
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity (= budget).

Use $i$-th budget (originally $\ell_j/(m' + 1)$) when $i - 1$ other jobs are active.
Our Algorithm

Open \( m' \) machines. Charge the delay of a job to its laxity (\( = \) budget).

Use \( i \)-th budget (originally \( \ell_j/(m' + 1) \)) when \( i - 1 \) other jobs are active.

1st budget empty, delay job, and charge 1st budget
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity (= budget).

Use $i$-th budget (originally $\ell_j/(m' + 1)$) when $i - 1$ other jobs are active.
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity ($= \text{budget}$).

Use $i$-th budget (originally $\ell_j/(m' + 1)$) when $i - 1$ other jobs are active.
Our Algorithm

Open $m'$ machines. Charge the delay of a job to its laxity ($= \text{budget}$).

Use $i$-th budget (originally $\ell_j/(m' + 1)$) when $i - 1$ other jobs are active.
To show: when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.
To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
Analysis Sketch

To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
- We construct a failure set \((F, T)\).

\[
\text{Lower Bound Theorem implies } m' = \mathcal{O}(m \log m) .
\]

Choosing \( m' = \mathcal{O}(m \log m) \) gives a contradiction.

Theorem Algorithm is \( \mathcal{O}(\log m) \)-competitive for online machine minimization.
To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
- We construct a failure set \((F, T)\).
Analysis Sketch

To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
- We construct a failure set \((F, T)\).

Job set \( F \)

Time points \( T \)  
\[ m' \text{-th budget was charged} \]
Analysis Sketch

To show: when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.

- Suppose some job $j^*$ has an empty $m'$-th budget.
- We construct a failure set $(F, T)$.

---

Job set F

- (m'−1)-th active job

Time points T

- m’-th budget was charged

(1/m'+1, 1)

Lemma: Failure set $(F, T)$ is a $(m'+1, 1/m'+1)$-critical pair.

→ Each $t \in T$ is covered by $\geq m' + 1$ distinct jobs from $F$.

→ Each $j \in F$ run out of budget during $T$, i.e., $|T \cap I(j)| \geq 1/m' + 1 \cdot \ell_j$.

Lower Bound Theorem implies $m' = \mathcal{O}(m \log m)$.

Choosing $m' = \mathcal{O}(m \log m)$ gives a contradiction.

Theorem: Algorithm is $O(\log m)$-competitive for online machine minimization.
To show: when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.

- Suppose some job $j^*$ has an empty $m'$-th budget.
- We construct a failure set $(F, T)$.

Job set $F$

Time points $T$

$(m' - 2)$-th budget was charged
Analysis Sketch

**To show:** when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.

- Suppose some job $j^*$ has an empty $m'$-th budget.
- We construct a failure set $(F, T)$.

![Diagram showing job set F and time points T with (m’–2)–th active jobs highlighted]
To show: when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.

- Suppose some job $j^*$ has an empty $m'$-th budget.
- We construct a failure set $(F, T)$.

### Lemma
Failure set $(F, T)$ is a $(m' + 1, 1)$-critical pair.

- Each $t \in T$ is covered by $\geq m' + 1$ distinct jobs from $F$.
- Each $j \in F$ runs out of budget during $T$, i.e., $|T \cap I(j)| \geq 1 \cdot m' + 1 \cdot \ell_j$.

Lower Bound Theorem implies $m' = \mathcal{O}(m \log m)$.

Choosing $m' = \mathcal{O}(m \log m)$ gives a contradiction.

Theorem: Algorithm is $\mathcal{O}(\log m)$-competitive for online machine minimization.
Analysis Sketch

To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
- We construct a failure set \((F, T)\).

Lemma: Failure set \((F, T)\) is a \((m' + 1, \frac{1}{m' + 1})\)-critical pair.
Analysis Sketch

To show: when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.

- Suppose some job $j^*$ has an empty $m'$-th budget.
- We construct a failure set $(F, T)$.

**Lemma:** Failure set $(F, T)$ is a $(m' + 1, \frac{1}{m'+1})$-critical pair.

- Each $t \in T$ is covered by $\geq m' + 1$ distinct jobs from $F$.
- Each $j \in F$ run out of budget during $T$, i.e., $|T \cap I(j)| \geq \frac{1}{m'+1} \cdot \ell_j$. 

Lower Bound Theorem implies $m' = \mathcal{O}(m \log m)$.

Choosing $m' = \mathcal{O}(m \log m)$ gives a contradiction.

Theorem: Algorithm is $O(\log m)$-competitive for online machine minimization.
Analysis Sketch

To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
- We construct a failure set \((F, T)\).

Lemma: Failure set \((F, T)\) is a \((m' + 1, \frac{1}{m'+1})\)-critical pair.

- Each \( t \in T \) is covered by \( \geq m' + 1 \) distinct jobs from \( F \).
- Each \( j \in F \) run out of budget during \( T \), i.e., \(|T \cap I(j)| \geq \frac{1}{m'+1} \cdot \ell_j\).

- Lower Bound Theorem implies \( m' = \mathcal{O}(m \log m') \).
Analysis Sketch

To show: when \( m' = \mathcal{O}(m \log m) \), there is no \((m' + 1)\)-th active job.

- Suppose some job \( j^* \) has an empty \( m' \)-th budget.
- We construct a failure set \((F, T)\).

**Lemma:** Failure set \((F, T)\) is a \((m' + 1, \frac{1}{m'+1})\)-critical pair.

\[
\rightarrow \text{ Each } t \in T \text{ is covered by } \geq m' + 1 \text{ distinct jobs from } F.
\rightarrow \text{ Each } j \in F \text{ run out of budget during } T, \text{ i.e., } |T \cap I(j)| \geq \frac{1}{m'+1} \cdot \ell_j.
\]

- Lower Bound Theorem implies \( m' = \mathcal{O}(m \log m') \).
- Choosing \( m' = \mathcal{O}(m \log m) \) gives a contradiction.
Analysis Sketch

To show: when $m' = \mathcal{O}(m \log m)$, there is no $(m' + 1)$-th active job.

- Suppose some job $j^*$ has an empty $m'$-th budget.
- We construct a failure set $(F, T)$.

Lemma: Failure set $(F, T)$ is a $(m' + 1, \frac{1}{m' + 1})$-critical pair.

- Each $t \in T$ is covered by $\geq m' + 1$ distinct jobs from $F$.
- Each $j \in F$ run out of budget during $T$, i.e., $|T \cap I(j)| \geq \frac{1}{m' + 1} \cdot \ell_j$.

- Lower Bound Theorem implies $m' = \mathcal{O}(m \log m')$.
- Choosing $m' = \mathcal{O}(m \log m)$ gives a contradiction.

Theorem

Algorithm is $\mathcal{O}(\log m)$-competitive for online machine minimization.
Two major special cases

Agreeable instances
for any two jobs $j$ and $k$:
$r_j < r_k$ implies $d_j \leq d_k$

Laminar instances
for any two intersecting time windows $I_j, I_k$: either $I_j \subseteq I_k$ or $I_k \subseteq I_j$
Two major special cases

Agreeable instances
for any two jobs $j$ and $k$:
$r_j < r_k$ implies $d_j \leq d_k$

Laminar instances
for any two intersecting time windows $I_j, I_k$: either $I_j \subseteq I_k$ or $I_k \subseteq I_j$

Theorem
Our algorithm is $O(1)$-competitive for agreeable and laminar instances.
Two major special cases

**Agreeable instances**
for any two jobs \(j\) and \(k\):
\[ r_j < r_k \text{ implies } d_j \leq d_k \]

**Laminar instances**
for any two intersecting time windows \(I_j, I_k\):
either \(I_j \subseteq I_k\) or \(I_k \subseteq I_j\)

---

**Theorem**
Our algorithm is \(O(1)\)-competitive for **agreeable** and **laminar** instances.

→ slightly modified lower bound and failure-set construction
Open questions

Does there exist a constant-competitive online algorithm?
– Decrease the gap between $\frac{5}{4}$ and $O(\log m)$.

Approximability of non-preemptive offline problem?
– $O(\sqrt{\log n \log \log n})$-approximation Chuzhoy et al. (FOCS 2004)

Fixed-parameter tractability/approximability?
– Parameters such as: $m$, laxity $\ell_j$, $\alpha$, $p_{max}$, or $p_{max}/p_{min}$
– Some first results Cieliebak et al. IFIP 2004 van Bevern, Niedermeier, Suchy arxiv 2016
Open questions

- Does there exist a constant-competitive \textit{online} algorithm?
Open questions

- Does there exist a constant-competitive online algorithm?
  - Decrease the gap between $5/4$ and $O(\log m)$.
Open questions

- Does there exist a constant-competitive online algorithm?
  - Decrease the gap between $5/4$ and $\mathcal{O}(\log m)$.
- Approximability of non-preemptive offline problem?
  - $\mathcal{O}(\sqrt{\frac{\log n}{\log \log n}})$-approximation

  Chuzhoy et al. (FOCS 2004)
Open questions

- Does there exist a constant-competitive online algorithm?
  - Decrease the gap between $5/4$ and $O(\log m)$.

- Approximability of non-preemptive offline problem?
  - $O(\sqrt{\log n / \log \log n})$-approximation
    Chuzhoy et al. (FOCS 2004)

- Fixed-parameter tractability/approximability?
Open questions

- Does there exist a constant-competitive online algorithm?
  - Decrease the gap between $5/4$ and $O(\log m)$.

- Approximability of non-preemptive offline problem?
  - $O(\sqrt{\frac{\log n}{\log \log n}})$-approximation by Chuzhoy et al. (FOCS 2004)

- Fixed-parameter tractability/approximability?
  - Parameters such as: $m$, laxity $\ell_j$, $\alpha$, $p_{\text{max}}$, or $p_{\text{max}}/p_{\text{min}}$
Open questions

- Does there exist a constant-competitive **online** algorithm?
  - Decrease the gap between $5/4$ and $\mathcal{O}(\log m)$.

- **Approximability of non-preemptive offline** problem?
  - $\mathcal{O}(\sqrt{\frac{\log n}{\log \log n}})$-approximation  
    Chuzhoy et al. (FOCS 2004)

- **Fixed-parameter** tractability/approximability?
  - Parameters such as: $m$, laxity $\ell_j$, $\alpha$, $p_{\text{max}}$, or $p_{\text{max}}/p_{\text{min}}$
  - some first results  
    Cieliebak et al. IFIP 2004
    van Bevern, Niedermeier, Suchy arxiv 2016