A Competitive Algorithm for Flow Time on Unrelated Machines with Rejection

Giorgio Lucarelli\textsuperscript{1} Benjamin Moseley\textsuperscript{2} Nguyen Kim Thang\textsuperscript{3} Abhinav Srivastav\textsuperscript{4} Denis Trystram\textsuperscript{1}

\textsuperscript{1} Université Grenoble-Alpes
\textsuperscript{2} Carnegie Mellon University
\textsuperscript{3} Université d’Evry
\textsuperscript{4} ENS-Paris & Université Paris-Dauphine

Workshop on New Challenges in Scheduling Theory, 2018
Problem definition

**Instance:** a set of $m$ unrelated machines $M$, a set of $n$ jobs $J$, and for each job $j \in J$:
- a machine-dependent processing time $p_{ij}$
- a release date $r_j$
- a weight $w_j$

**Goal:** a non-preemptive schedule that minimizes the weighted flow time:

$$\sum_{j \in J} w_j (C_j - r_j)$$

where $C_j$ is the completion time of job $j \in J$
Problem definition

**Instance:** a set of $m$ unrelated machines $\mathcal{M}$, a set of $n$ jobs $\mathcal{J}$, and for each job $j \in \mathcal{J}$:
- a machine-dependent *processing time* $p_{ij}$
- a *release date* $r_j$
- a *weight* $w_j$

**Goal:** a non-preemptive schedule that minimizes the weighted flow time:

$$\sum_{j \in \mathcal{J}} w_j (C_j - r_j)$$

where $C_j$ is the completion time of job $j \in \mathcal{J}$

**Setting**
- jobs arrive *online*
- job characteristics $(w_j, p_{ij})$ become known after the release of $j$
Performance guarantees

**Competitive ratio** of an *online* algorithm $B$ is defined as:

$$\max_{\mathcal{I} \in S} \frac{\text{Obj value of } B \text{ on instance } \mathcal{I}}{\text{Obj value of offline OPT on instance } \mathcal{I}}$$

where $S$ is set of all possible instances
Previous work for non-preemptive scheduling

Offline

- $\Omega(n^{1/2-\epsilon})$ on a single machine for unweighted flow time [Kellerer et al. 1999]
- $O(\sqrt{\frac{n}{m}} \log \frac{n}{m})$-approximation algorithm for identical machines [Leonardi & Raz 2007]

Online

- $\Omega(n)$ on a single machine for unweighted flow time [Chekuri et al. 2001]
- $\Theta(\frac{p_{\text{max}}}{p_{\text{min}}} + 1)$-competitive algorithm for a single machine [Tao and Liu 2013]
The algorithm is allowed to use more resources than the optimal

- use higher speed [Phillips et al. 1997, Kalyanasundaram and Pruhs 2000]
- use more machines [Phillips et al. 1997]
Resource augmentation

- The algorithm is allowed to use more resources than the optimal
  - use higher speed [Phillips et al. 1997, Kalyanasundaram and Pruhs 2000]
  - use more machines [Phillips et al. 1997]
  - reject jobs [Choudhury et al. 2015]
Resource augmentation

- The algorithm is allowed to use more resources than the optimal
  - use higher speed [Phillips et al. 1997, Kalyanasundaram and Pruhs 2000]
  - use more machines [Phillips et al. 1997]
  - reject jobs [Choudhury et al. 2015]

- Refined competitive ratio:

\[
\frac{\text{algorithm's solution using resource augmentation}}{\text{offline optimal solution (without resource augmentation)}}
\]
Previous work (cont’d)

Offline

- 12-speed 4-approximation algorithm for a single machine [Bansal et al. 2007]
- \((1 + \epsilon)\)-speed \((1 + \epsilon)\)-approximation quasi-polynomial time algorithm for identical machines [Im et al. 2015]

Online

- \(O(\log \frac{p_{\text{max}}}{p_{\text{min}}})\)-machines \(O(1)\)-competitive for identical machines [Phillips et al. 1997]
- \(O(\log n)\)-machine \(O(1)\)-speed \(O(1)\)-competitive for total (unweighted) flow time on identical machines [Phillips et al. 1997]
- \(\ell\)-machines \(O(\min\{\ell \sqrt{\frac{p_{\text{max}}}{p_{\text{min}}}}, \sqrt{n}\})\)-competitive algorithm for total (unweighted) flow time on a single machine [Epstein and van Stee 2006]
  - optimal up to a constant factor for constant \(\ell\)
Lower bound with speed augmentation: for any speed augmentation $s \leq 10\sqrt{\frac{p_{\text{max}}}{p_{\text{min}}}}$, every deterministic algorithm has competitive ratio at least $\Omega(\frac{10\sqrt{p_{\text{max}}}}{p_{\text{min}}})$ on a single machine [Lucarelli et al. 2016]
Previous work (cont’d)

Lower bound with speed augmentation: for any speed augmentation $s \leq 10\sqrt{\frac{p_{max}}{p_{min}}}$, every deterministic algorithm has competitive ratio at least $\Omega(10\sqrt{\frac{p_{max}}{p_{min}}})$ on a single machine [Lucarelli et al. 2016]

Speed + Rejection

- $\epsilon_s$-speed, $\epsilon_r$-rejection $O\left(\frac{1}{\epsilon_s \cdot \epsilon_r}\right)$-competitive algorithm for the weighted flow time problem [Lucarelli et al. 2016]
Our contributions

New Result

- $O(\epsilon)$-rejection $O\left(\frac{1}{\epsilon^3}\right)$-competitive algorithm for the weighted flow time problem in the non-preemptive setting
Our contributions

New Result

- $O(\epsilon)$-rejection $O\left(\frac{1}{\epsilon^3}\right)$-competitive algorithm for the **weighted flow time problem** in the non-preemptive setting

In this talk, I sketch the proof for minimizing the **total (unweighted) flow time**

$O(\epsilon)$-rejection $O\left(\frac{1}{\epsilon^2}\right)$-competitive algorithm for the **flow time problem**
Variable

\[
    x_{ij}(t) = \begin{cases} 
        1, & \text{if job } j \text{ is executed on machine } i \text{ at time } t \\
        0, & \text{otherwise} 
    \end{cases}
\]
Linear programming formulation

Variable

\[ x_{ij}(t) = \begin{cases} 
1, & \text{if job } j \text{ is executed on machine } i \text{ at time } t \\
0, & \text{otherwise} 
\end{cases} \]

Lower bounds on flow time objective

- **processing time of job** \( j \):
  \[ p_{ij} = \int_{r_j}^\infty x_{ij}(t) \, dt \]
Linear programming formulation

Variable

- \( x_{ij}(t) = \begin{cases} 
1, & \text{if job } j \text{ is executed on machine } i \text{ at time } t \\
0, & \text{otherwise} 
\end{cases} \)

Lower bounds on flow time objective

- **processing time of job** \( j \):
  \[ p_{ij} = \int_{r_j}^{\infty} x_{ij}(t) \, dt \]

- **fractional flow time of job** \( j \):
  \[ \int_{r_j}^{\infty} \frac{q_{ij}(t)}{p_{ij}} \, dt = \int_{r_j}^{\infty} \frac{(t - r_j)}{p_{ij}} x_{ij}(t) \, dt \]

\( (q_{ij}(t) \text{: remaining processing time of } j \text{ at } t) \)
Linear programming formulation

Variable

\[ x_{ij}(t) = \begin{cases} 
1, & \text{if job } j \text{ is executed on machine } i \text{ at time } t \\
0, & \text{otherwise} 
\end{cases} \]

Lower bounds on flow time objective

- processing time of job \( j \):
  \[ p_{ij} = \int_{r_j}^{\infty} x_{ij}(t) \, dt \]

- fractional flow time of job \( j \):
  \[ \int_{r_j}^{\infty} \frac{q_{ij}(t)}{p_{ij}} \, dt = \int_{r_j}^{\infty} \frac{(t - r_j)}{p_{ij}} x_{ij}(t) \, dt \]

\( (q_{ij}(t): \text{remaining processing time of } j \text{ at } t) \)
Linear programming relaxation

**Primal**

\[
\begin{align*}
\min & \quad \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}^p}{p_{ij}} x_{ij}(t) dt \\
\sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt & \geq 1 \quad \forall j \in \mathcal{J} \\
\sum_{j \in \mathcal{J}} x_{ij}(t) & \leq 1 \quad \forall i \in \mathcal{M}, t \geq 0 \\
x_{ij}(t) & \geq 0 \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0
\end{align*}
\]
Linear programming relaxation

**Primal**

\[
\begin{align*}
\min & \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}}{p_{ij}} x_{ij}(t) dt \\
\sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt & \geq 1 \quad \forall j \in \mathcal{J} \\
\sum_{j \in \mathcal{J}} x_{ij}(t) & \leq 1 \quad \forall i \in \mathcal{M}, t \geq 0 \\
x_{ij}(t) & \geq 0 \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0
\end{align*}
\]

**Dual**

\[
\begin{align*}
\max & \sum_{j \in \mathcal{J}} \lambda_j - \sum_{i \in \mathcal{M}} \int_{0}^{\infty} \gamma_i(t) dt \\
\frac{\lambda_j}{p_{ij}} - \gamma_i(t) & \leq \frac{t - r_j + p_{ij}}{p_{ij}} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq r_j \\
\lambda_j, \gamma_i(t) & \geq 0 \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0
\end{align*}
\]
Rejection interpretation

Primal

\[ \min \sum_{i \in M} \sum_{j \in J \setminus \mathcal{R}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}}{p_{ij}} x_{ij}(t) dt \]

\[ \sum_{i \in M} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \quad \forall j \in J \setminus \mathcal{R} \]

\[ \sum_{j \in J \setminus \mathcal{R}} x_{ij}(t) \leq 1 \quad \forall i \in M, t \geq 0 \]

\[ x_{ij}(t) \geq 0 \quad \forall i \in M, j \in J \setminus \mathcal{R}, t \geq 0 \]

Dual

\[ \max \sum_{j \in J} \lambda_j - \sum_{i \in M} \int_{0}^{\infty} \gamma_i(t) dt \]

\[ \frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \frac{t - r_j + p_{ij}}{p_{ij}} \quad \forall i \in M, j \in J, t \geq r_j \]

\[ \lambda_j, \gamma_i(t) \geq 0 \quad \forall i \in M, j \in J, t \geq 0 \]
Intuition of rejection

![Diagram showing time from 0 to P]
Intuition of rejection

A. Srivastav

Flow Time

2018 12 / 21
Intuition of rejection
Intuition of rejection

A. Srivastav

Flow Time 2018 12 / 21
Intuition of rejection
Intuition of rejection

- \( P \) small jobs
- each small job has flow time \( P \)
Intuition of rejection

- \( P \) small jobs
- each small job has flow time \( P \)
- ... while in the optimal it has flow time 1
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
$P$ small jobs

each small job has flow time $P$

... while in the optimal it has flow time 1

but we can reject ...
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
Intuition of rejection

- $P$ small jobs
- each small job has flow time $P$
- ... while in the optimal it has flow time 1
- but we can reject ...
Rejection policy for non-preemption

- \( \epsilon \in (0, 1) \): the rejection constant

\[ \epsilon \in (0, 1) \]: the rejection constant
Rejection policy for non-preemption

- $\epsilon \in (0, 1)$: the rejection constant

1. At the beginning of the execution of job $k$ on machine $i$
   $\Rightarrow$ introduce a counter $v_k = 0$

2. Each time a job $j$, with $p_{ij} < p_{ik}$, arrives during the execution of $k$ and $j$ is dispatched to machine $i$
   $\Rightarrow v_k \leftarrow v_k + 1$

3. Interrupt and reject $k$ the first time where $v_k \geq \frac{1}{\epsilon}$
Rejection policy for non-preemption

- $\epsilon \in (0, 1)$: the rejection constant

1. At the beginning of the execution of job $k$ on machine $i$
   $\Rightarrow$ introduce a counter $v_k = 0$

2. Each time a job $j$, with $p_{ij} < p_{ik}$, arrives during the execution of $k$
   and $j$ is dispatched to machine $i$
   $\Rightarrow v_k \leftarrow v_k + 1$

3. Interrupt and reject $k$ the first time where $v_k \geq \frac{1}{\epsilon}$

In the weighted case, interrupt and reject $k$ the first time when $v_k \geq \frac{w_k}{\epsilon}$
Second rejection policy
Second rejection policy

A. Srivastav
Flow Time
2018 14 / 21
Second rejection policy

A. Srivastav

Flow Time

2018 14 / 21
Second rejection policy
Second rejection policy
Second rejection policy

- Reject the last job in the queue
- Each job in the queue has higher priority than the rejected job
Second rejection policy

- Reject the last job in the queue
- Each job in the queue has higher priority than the rejected job
Second rejection policy

- Reject the last job in the queue
- Each job in the queue has higher priority than the rejected job
Second rejection policy

- Reject the last job in the queue
- Each job in the queue has higher priority than the rejected job
- We again reject the last job in the queue.
Second rejection policy

- Reject the last job in the queue
- Each job in the queue has higher priority than the rejected job
- We again reject the last job in the queue.
Reject the last job in the queue
Each job in the queue has higher priority than the rejected job
We again reject the last job in the queue.
Second rejection policy

- Reject the last job in the queue.
- Each job in the queue has higher priority than the rejected job.
- We again reject the last job in the queue.
- Each rejected job can be mapped to at most $1/\epsilon$-jobs in the queue.
- Rejected jobs complete later than jobs in the queue.
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in \mathcal{M}$
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in M$
2. Each time a job is dispatched to a machine $i$, $c_i$ is incremented by 1
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in \mathcal{M}$
2. Each time a job is dispatched to a machine $i$, $c_i$ is incremented by 1
3. Reject the job with the smallest priority when $c_i = \frac{1}{\epsilon} + 1$
4. Re-initialize $c_i$ to 0.
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in M$
2. Each time a job is dispatched to a machine $i$, $c_i$ is incremented by 1
3. Reject the job with the smallest priority when $c_i = \frac{1}{\epsilon} + 1$
4. Re-initialize $c_i$ to 0.
5. Rejected jobs can pay $\epsilon$-factor more for the future arriving jobs
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in \mathcal{M}$
2. Each time a job is dispatched to a machine $i$, $c_i$ is incremented by 1
3. Reject the job with the smallest priority when $c_i = \frac{1}{\epsilon} + 1$
4. Re-initialize $c_i$ to 0.
5. Rejected jobs can pay $\epsilon$-factor more for the future arriving jobs

In the weighted case, we reject the smallest density job $\ell$ as soon as $c_i \geq \frac{1}{\epsilon} w_\ell$
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in \mathcal{M}$
2. Each time a job is dispatched to a machine $i$, $c_i$ is incremented by 1
3. Reject the job with the smallest priority when $c_i = \frac{1}{\epsilon} + 1$
4. Re-initialize $c_i$ to 0.
5. Rejected jobs can pay $\epsilon$-factor more for the future arriving jobs

In the \textit{weighted case}, we reject the smallest density job $\ell$ as soon as $c_i \geq \frac{1}{\epsilon} w_\ell$

We reject more jobs in the weighted case to get bound in the dual objective
Second rejection policy

1. At time $t = 0$ introduce $c_i = 0$ for every machine $i \in M$
2. Each time a job is dispatched to a machine $i$, $c_i$ is incremented by 1
3. Reject the job with the smallest priority when $c_i = \frac{1}{\epsilon} + 1$
4. Re-initialize $c_i$ to 0.
5. Rejected jobs can pay $\epsilon$-factor more for the future arriving jobs

In the weighted case, we reject the smallest density job $\ell$ as soon as $c_i \geq \frac{1}{\epsilon} w_\ell$

We reject more jobs in the weighted case to get bound in the dual objective

**Lemma:** We reject at most an $O(\epsilon)$-fraction of the jobs
Scheduling policy
Scheduling policy

A. Srivastav

Flow Time 2018 16 / 21
For each machine $i$

⇒ schedule the jobs dispatched on $i$ in Shortest Processing Time order
Scheduling policy

Marginal increase

- $A_1$: set of jobs with smaller processing time than $j$
  - contribute to the flow time of the new job $j$
- $A_2$: set of jobs with bigger processing time than $j$
  - the new job $j$ delay them by $p_{ij}$
Marginal increase

\[
\Delta_{ij} = \begin{cases} 
(p_{ik}(r_j) + \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell}) + |A_2| \cdot p_{ij} & \text{if } k \text{ is not rejected} \\
\sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + \left(|A_2| \cdot p_{ij} - |A_1 \cup A_2| \cdot p_{ik}(r_j)\right) & \text{otherwise}
\end{cases}
\]
Charging marginal increase

Marginal increase

\[
\Delta_{ij} \leq \begin{cases} 
    p_{ik}(r_j) + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } k \text{ is not rejected} \\
    \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise}
\end{cases}
\]
Charging marginal increase

Marginal increase

\[
\Delta_{ij} \leq \begin{cases} 
  p_{ik}(r_j) + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } k \text{ is not rejected} \\
  \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise}
\end{cases}
\]

Recall rejection: increase the counter of \( k \) only if \( j \) has smaller processing time
Charging marginal increase

Marginal increase

\[
\Delta_{ij} \leq \begin{cases} 
  p_{ik}(r_j) + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } k \text{ is not rejected} \\
  \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise}
\end{cases}
\]

Recall rejection: increase the counter of \(k\) only if \(j\) has smaller processing time

Define:

\[
\lambda_{ij} = \begin{cases} 
  \frac{1}{\epsilon_r} p_{ij} + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } p_{ij} < p_{ik} \\
  \frac{1}{\epsilon_r} p_{ij} + p_{ik}(r_j) + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise}
\end{cases}
\]
Charging marginal increase

Marginal increase

\[
\Delta_{ij} \leq \begin{cases} 
    p_{ik}(r_j) + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } k \text{ is not rejected} \\
    \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise}
\end{cases}
\]

Recall rejection: increase the counter of \( k \) only if \( j \) has smaller processing time

Define:

\[
\lambda_{ij} = \begin{cases} 
    \frac{1}{\epsilon_r} p_{ij} + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } p_{ij} < p_{ik} \\
    \frac{1}{\epsilon_r} p_{ij} + p_{ik}(r_j) + \left( \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise}
\end{cases}
\]

prediction term
Dispatching policy

- **Immediate dispatch** at arrival and never change this decision
- Dispatch $j$ to the machine $i$ of minimum $\lambda_{ij}$
Dual variables

- $\lambda_j = \frac{\epsilon}{1+\epsilon} \min_i \lambda_{ij}$
- $\gamma_i(t) = \frac{\epsilon}{(1+\epsilon)^2} (|U_i(t)| + |R_i(t)|)$ *new technique*
Dual variables

- \( \lambda_j = \frac{\epsilon}{1+\epsilon} \min_i \lambda_{ij} \)
- \( \gamma_i(t) = \frac{\epsilon}{(1+\epsilon)^2} (|U_i(t)| + |R_i(t)|) \) new technique

Recall dual objective

\[
\sum_{j \in \mathcal{J}} \lambda_j - \sum_{i \in \mathcal{M}} \int_0^\infty \gamma_i(t) dt
\]
Dual variables

\[ \lambda_j = \frac{\epsilon}{1 + \epsilon} \min_i \lambda_{ij} \]

\[ \gamma_i(t) = \frac{\epsilon}{(1 + \epsilon)^2} \left( |U_i(t)| + |R_i(t)| \right) \text{ new technique} \]

Recall dual objective

\[ \sum_{j \in J} \lambda_j - \sum_{i \in M} \int_0^\infty \gamma_i(t) dt \geq \text{total marginal increase} \]

\[ = \text{total flow time} \]
Dual variables

- \( \lambda_j = \frac{\epsilon}{1+\epsilon} \min_i \lambda_{ij} \)

- \( \gamma_i(t) = \frac{\epsilon}{(1+\epsilon)^2} (|U_i(t)| + |R_i(t)|) \) new technique

Recall dual objective

\[
\sum_{j \in J} \lambda_j - \sum_{i \in M} \int_0^\infty \gamma_i(t) dt \geq \text{total marginal increase} = \text{total flow time} = \frac{\text{total flow time}}{1+\epsilon}
\]
Putting all together

- **first rejection**: update the counter of executed job when a new job arrives
  \[ \Rightarrow \text{reject if the counter exceeds a threshold based on } \epsilon \]

- **second rejection**: update the counter of machine where a new job is dispatched
  \[ \Rightarrow \text{reject if the counter exceeds a threshold based on } \epsilon \]

- **immediate dispatch**: based on minimum \( \lambda_{ij} \)

- **schedule**: select the pending job of smallest processing time

**Theorem:** \( O(\epsilon) \)-rejection \( O(1/\epsilon^2) \)-competitive algorithm for the total flow time problem

**Proof:**
- Compare primal with dual objectives
- Prove feasibility of dual constraint
- Rejection is bounded by \( \epsilon \)
Putting all together

- **first rejection**: update the counter of executed job when a new job arrives
  ⇒ reject if the counter exceeds a threshold based on $\epsilon$

- **second rejection**: update the counter of machine where a new job is dispatched
  ⇒ reject if the counter exceeds a threshold based on $\epsilon$

- **immediate dispatch**: based on minimum $\lambda_{ij}$

- **schedule**: select the pending job of smallest processing time

**Theorem**: $O(\epsilon)$-rejection $O\left(\frac{1}{\epsilon^2}\right)$-competitive algorithm for the total flow time problem

**Proof**:

- Compare primal with dual objectives
- Prove feasibility of dual constraint
- Rejection is bounded by $\epsilon$
Concluding remarks

- Power of rejections!
- Non-preemptive results with rejection only
- Scalable algorithms

Open Problem: Is there a model where one can make immediate decisions?

Thank you!
Concluding remarks

- Power of rejections!
- Non-preemptive results with rejection only
- Scalable algorithms

Open Problem: Is there a model where one can make immediate decisions?
Concluding remarks

- Power of rejections!
- Non-preemptive results with rejection only
- Scalable algorithms

Open Problem: Is there a model where one can make immediate decisions?

Thank you!