Online Non-preemptive Scheduling in a Resource Augmentation Model based on Duality

Giorgio Lucarelli[1] Nguyen Kim Thang[2] Abhinav Srivastav[1]
Denis Trystram[1]

[1]LIG, University of Grenoble-Alpes
[2]IBISC, University of Evry Val d'Essonne

New Challenges in Scheduling Theory, 2016

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 total weighted flow time:

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

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

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 total 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 (p_{ij}, w_j) become known after the release of j

Previous work for non-preemptive scheduling

Offline

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

Online

- Lower bound: $\Omega(n)$ even on a single machine for total (unweighted) flow time [Chekuri et al. 2001]
- $\Theta(\frac{p_{\max}}{p_{\min}}+1)$-competitive algorithm for a single machine [Tao and Liu 2013]

Resource augmentation

- The algorithm is allowed to use more resources than the optimal
  - use highest 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 highest 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 highest 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

- $O(\log \frac{p_{\max}}{p_{\min}})$-machines $O(1)$-competitive for identical machines [Phillips et al. 1997]
- $O(\log n)$-machine $O(1)$-speed 1-competitive for total (unweighted) flow time on identical machines [Phillips et al. 1997]
- ℓ-machines $O(\min\{\sqrt[\ell]{\frac{p_{\max}}{p_{\min}}}, \sqrt[\ell]{n}\})$-competitive algorithm for total (unweighted) flow time on a single machine [Epstein and van Stee 2006]
  - optimal up to a constant factor for constant ℓ

Our contribution

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

Our contribution

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

Resource augmentation algorithms

- $(1+\epsilon_s)$-speed ϵ_r-rejection $\frac{2(1+\epsilon_r)(1+\epsilon_s)}{\epsilon_r \epsilon_s}$-competitive algorithm
- extension for ℓ_k-norms

Linear programming formulation

Definitions

- $\delta_{ij} = \frac{w_j}{p_{ij}}$: *density* of the job j on machine i
- $\mathcal{R}$: set of rejected jobs
- 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

Definitions

- $\delta_{ij} = \frac{w_j}{p_{ij}}$: *density* of the job j on machine i
- $\mathcal{R}$: set of rejected jobs
- 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 to our objective

- fractional flow time of job j:

$$\int_{r_j}^{\infty} \delta_{ij}(t - r_j)x_{ij}(t)dt$$

- weighted processing time of job j

$$w_j p_j = w_j \int_{r_j}^{\infty} x_{ij}(t)dt = \int_{r_j}^{\infty} \delta_{ij} p_{ij} x_{ij}(t)dt$$

Linear programming relaxation

Primal

$$\min \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \delta_{ij}(t - r_j + p_{ij}) x_{ij}(t) dt$$

$$\sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_{ij}(t) \leq 1 \qquad \forall i \in \mathcal{M}, t \geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0$$

Linear programming relaxation

Primal

$$\min \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}} \int_{r_j}^{\infty} \delta_{ij}(t - r_j + p_{ij}) x_{ij}(t) dt$$

$$\sum_{i\in\mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}$$

$$\sum_{j\in\mathcal{J}} x_{ij}(t) \leq 1 \qquad \forall i \in \mathcal{M}, t \geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0$$

Dual

$$\max \sum_{j\in\mathcal{J}} \lambda_j - \sum_{i\in\mathcal{M}} \int_0^{\infty} \gamma_i(t) dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \delta_{ij}(t - r_j + p_{ij}) \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0$$

Speed interpretation

Primal

$$\min \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \delta_{ij}(t - r_j + p_{ij}) x_{ij}(t) dt$$

$$\sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_{ij}(t) \leq \frac{1}{1+\epsilon_s} \qquad \forall i \in \mathcal{M}, t \geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0$$

Dual

$$\max \sum_{j \in \mathcal{J}} \lambda_j - \frac{1}{1+\epsilon_s} \sum_{i \in \mathcal{M}} \int_0^{\infty} \gamma_i(t) dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \delta_{ij}(t - r_j + p_{ij}) \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0$$

Rejection interpretation

Primal

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

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

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

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

Dual

$$\max \sum_{j \in \mathcal{J} \setminus \mathcal{R}} \lambda_j - \sum_{i \in \mathcal{M}} \int_{0}^{\infty} \gamma_i(t) dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \delta_{ij}(t - r_j + p_{ij}) \qquad \forall i \in \mathcal{M}, j \in \mathcal{J} \setminus \mathcal{R}, t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J} \setminus \mathcal{R}, t \geq 0$$

Competitive ratio

Primal

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

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

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

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

Dual

$$\max \sum_{j \in \mathcal{J}} \lambda_j - \frac{1}{1+\epsilon_s} \sum_{i \in \mathcal{M}} \int_0^{\infty} \gamma_i(t) dt$$

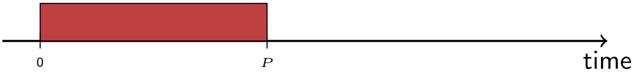
$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \delta_{ij}(t - r_j + p_{ij}) \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0$$

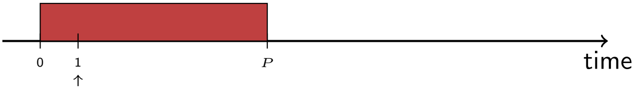
Competitive ratio

$$\frac{\text{Primal}(\text{speed} = 1, \mathcal{J} \setminus \mathcal{R})}{\text{Dual}(\text{speed} = \frac{1}{1+\epsilon_s}, \mathcal{J})} = \frac{\displaystyle\sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J} \setminus \mathcal{R}} \int_{r_j}^{\infty} \delta_{ij}(t - r_j + p_{ij}) x_{ij}(t) dt}{\displaystyle\sum_{j \in \mathcal{J}} \lambda_j - \frac{1}{1+\epsilon_s} \sum_{i \in \mathcal{M}} \int_0^{\infty} \gamma_i(t) dt}$$

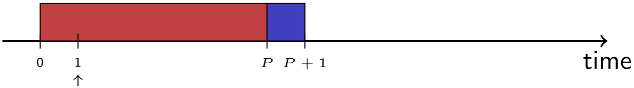
Intuition of rejection



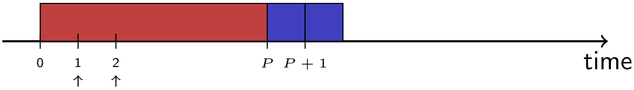
Intuition of rejection



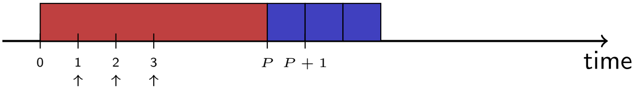
Intuition of rejection



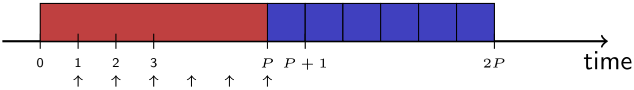
Intuition of rejection



Intuition of rejection

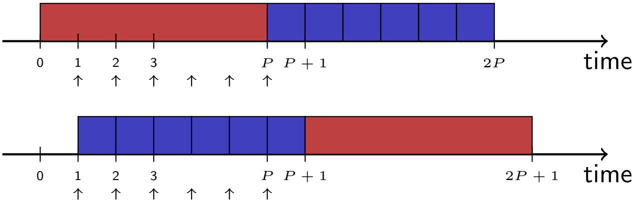


Intuition of rejection



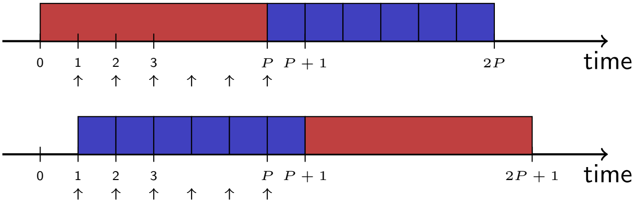
- P small jobs
- each small job has flow time P

Intuition of rejection



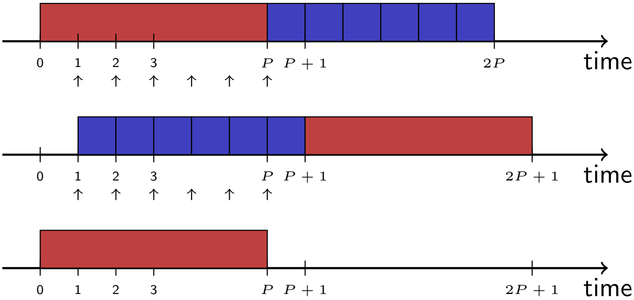
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1

Intuition of rejection



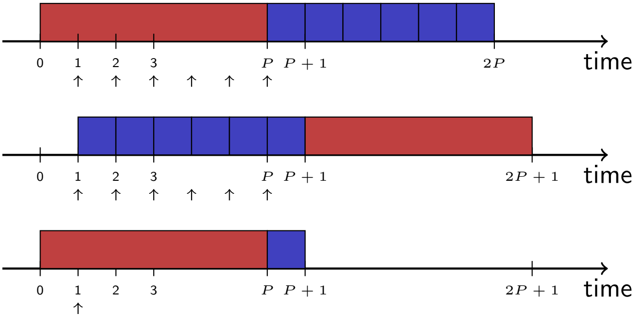
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1
- but we can reject ...

Intuition of rejection



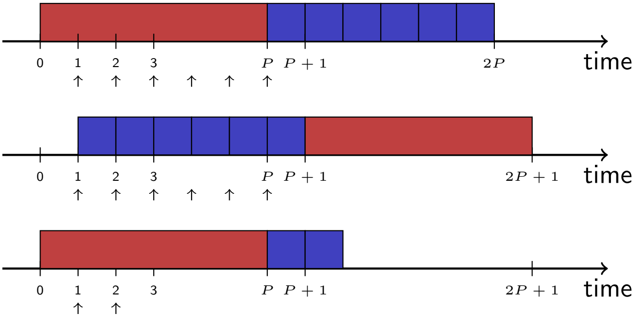
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1
- but we can reject ...

Intuition of rejection



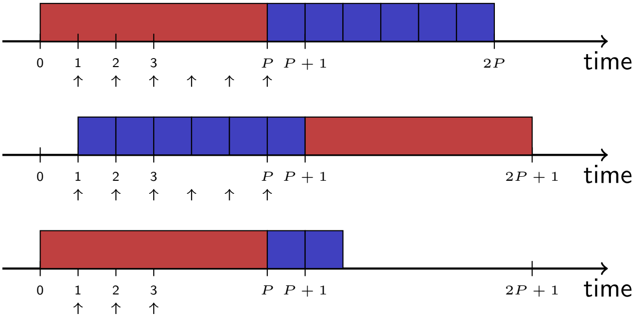
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1
- but we can reject ...

Intuition of rejection



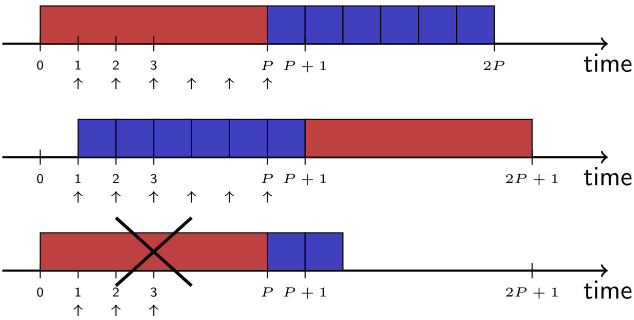
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1
- but we can reject ...

Intuition of rejection



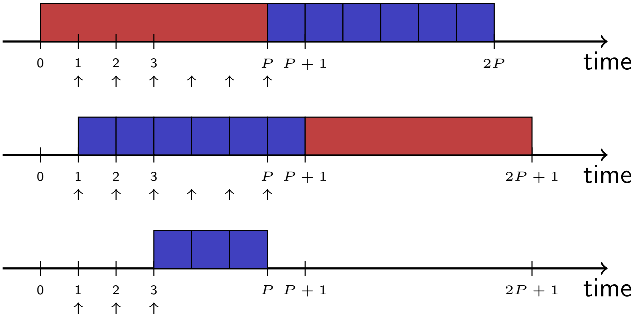
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1
- but we can reject ...

Intuition of rejection



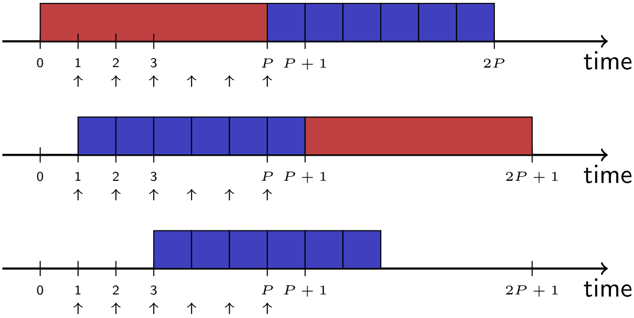
- P small jobs
- each small job has flow time P
- ... while in the optimal it has flow time 1
- but we can reject ...

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 ...

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

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

Rejection policy

- $\epsilon_r \in (0,1)$: 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 $\frac{w_j}{p_{ij}} > \frac{w_k}{p_{ik}}$, arrives during the execution of k and j is dispatched to machine i
   $\Rightarrow v_k \leftarrow v_k + w_j$
3. Interrupt and reject k the first time where $v_k \geq \frac{w_k}{\epsilon_r}$

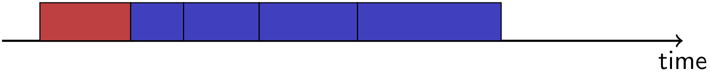
Rejection policy

- $\epsilon_r \in (0,1)$: 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 $\frac{w_j}{p_{ij}} > \frac{w_k}{p_{ik}}$, arrives during the execution of k and j is dispatched to machine i
$\Rightarrow v_k \leftarrow v_k + w_j$
3. Interrupt and reject k the first time where $v_k \geq \frac{w_k}{\epsilon_r}$

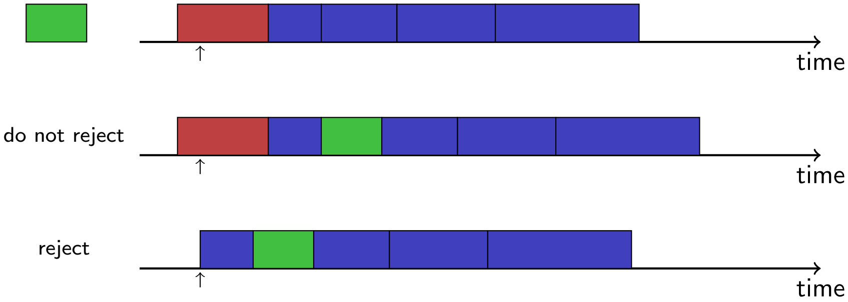
Lemma: We reject jobs with weight at most an ϵ_r-fraction of the total weight

Scheduling policy



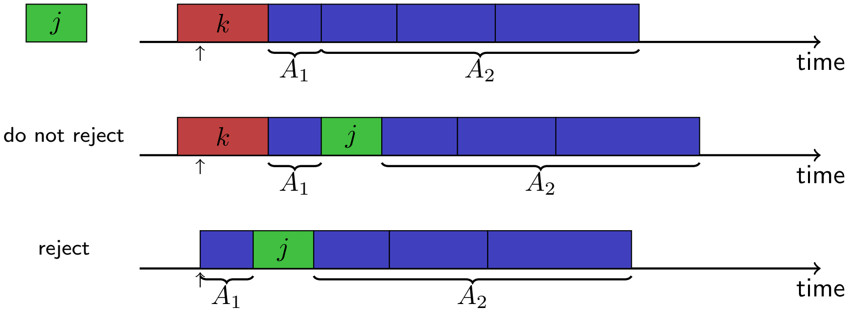
Scheduling policy

Scheduling policy



- For each machine i
  $\Rightarrow$ schedule the jobs dispatched on i in non-increasing order of density

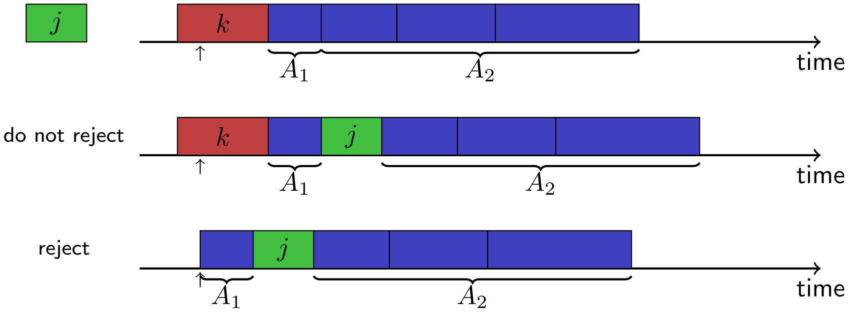
Scheduling policy



Marginal increase

- A_1: set of jobs with higher density than j
  - contribute to the flow time of the new job j
- A_2: set of jobs with lower density than j
  - the new job j delay them by p_{ij}

Scheduling policy



Marginal increase

$$\Delta_{ij} = \begin{cases} w_j \left(p_{ik}(r_j) + \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} \right) + p_{ij} \sum_{\ell \in A_2} w_\ell & \text{if } k \text{ is not rejected} \\ w_j \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + \left(p_{ij} \sum_{\ell \in A_2} w_\ell - p_{ik}(r_j) \sum_{\ell \in A_1 \cup A_2} w_\ell \right) & \text{otherwise} \end{cases}$$

Charging marginal increase

Marginal increase

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

Charging marginal increase

Marginal increase

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

Recall rejection: increase the counter of k only if j has biggest density

Charging marginal increase

Marginal increase

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

Recall rejection: increase the counter of k only if j has biggest density

Define:

$$\lambda_{ij} = \begin{cases} \frac{w_j}{\epsilon_r} p_{ij} + \left(w_j \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + p_{ij} \sum_{\ell \in A_2} w_\ell \right) & \text{if } \delta_{ij} > \delta_{ik} \\ \frac{w_j}{\epsilon_r} p_{ij} + w_j p_{ik}(r_j) + \left(w_j \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + p_{ij} \sum_{\ell \in A_2} w_\ell \right) & \text{otherwise} \end{cases}$$

Charging marginal increase

Marginal increase

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

Recall rejection: increase the counter of k only if j has biggest density

Define:

$$\lambda_{ij} = \begin{cases} \boxed{\frac{w_j}{\epsilon_r} p_{ij}} + \left(w_j \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + p_{ij} \sum_{\ell \in A_2} w_\ell \right) & \text{if } \delta_{ij} > \delta_{ik} \\ \boxed{\frac{w_j}{\epsilon_r} p_{ij}} + w_j p_{ik}(r_j) + \left(w_j \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + p_{ij} \sum_{\ell \in A_2} w_\ell \right) & \text{otherwise} \end{cases}$$

prediction term

Dispatching policy

- Immediate dispatch at arrival and never change this decision
- Dispatch j to the machine i of minimum λ_{ij}

Dual variables

- $\lambda_j = \min_i \lambda_{ij}$
- $\gamma_i(t) =$ weight of pending jobs on machine i

Dual variables

- $\lambda_j = \min_i \lambda_{ij}$
- $\gamma_i(t) =$ weight of pending jobs on machine i

Recall dual objective

$$\sum_{j \in \mathcal{J}} \lambda_j - \frac{1}{1+\epsilon_s} \sum_{i \in \mathcal{M}} \int_0^\infty \gamma_i(t) dt$$

Dual variables

- $\lambda_j = \min_i \lambda_{ij}$
- $\gamma_i(t) =$ weight of pending jobs on machine i

Recall dual objective

$$\boxed{\sum_{j \in \mathcal{J}} \lambda_j} - \frac{1}{1+\epsilon_s} \sum_{i \in \mathcal{M}} \int_0^\infty \gamma_i(t) dt$$

$\geq$ total marginal increase
$=$ total weighted flow time

Dual variables

- $\lambda_j = \min_i \lambda_{ij}$
- $\gamma_i(t) =$ weight of pending jobs on machine i

Recall dual objective

$$\boxed{\sum_{j \in \mathcal{J}} \lambda_j} - \frac{1}{1+\epsilon_s} \boxed{\sum_{i \in \mathcal{M}} \int_0^\infty \gamma_i(t) dt}$$

$\geq$ total marginal increase
$=$ total weighted flow time

$=$ total weighted flow time

Putting all together

- rejection: update the counter of executed job when a new job arrives
  $\Rightarrow$ reject if the counter exceeds a threshold based on ϵ_r
- immediate dispatch: based on minimum λ_{ij}
- schedule: select the pending job of highest density

Putting all together

- rejection: update the counter of executed job when a new job arrives
  $\Rightarrow$ reject if the counter exceeds a threshold based on ϵ_r
- immediate dispatch: based on minimum λ_{ij}
- schedule: select the pending job of highest density

Theorem: $(1+\epsilon_s)$-speed ϵ_r-rejection $\frac{2(1+\epsilon_r)(1+\epsilon_s)}{\epsilon_r \epsilon_s}$-competitive algorithm

Proof:

- Compare primal with dual objectives
- Prove feasibility of dual constraint
- Rejection is bounded by ϵ_r

ℓ_k-norm

- Use exactly the same policies
- More complicated analysis for dual feasibility

Theorem: $(1 + \epsilon_s)$-speed ϵ_r-rejection $O\left(\frac{k^{(k+3)/k}}{\epsilon_r^{1/k}\epsilon_s^{(k+2)/k}}\right)$-competitive algorithm

Concluding remarks

- power of rejections !
- Non-preemptive results with rejection + speed-augmentation
- Scalable algorithms

Concluding remarks

- power of rejections !
- Non-preemptive results with rejection + speed-augmentation
- Scalable algorithms

Question: Can we remove speed-augmentation ?

Concluding remarks

- power of rejections !
- Non-preemptive results with rejection + speed-augmentation
- Scalable algorithms

Question: Can we remove speed-augmentation ?

Generalized resource augmentation in conjunction with a duality-based approach

- unifies the existing models
- can introduce different/new models

Generalized resource augmentation

Definition

- Consider an optimization problem that can be formalized by a mathematical program.
- Let $\mathcal{P}$ be the set of feasible solutions of the program and let $\mathcal{Q} \subset \mathcal{P}$.
- Performance of an algorithm

$$\frac{\text{algorithm's solution over } \mathcal{P}}{\text{offline optimal solution over } \mathcal{Q}}$$

Generalized resource augmentation

Definition

- Consider an optimization problem that can be formalized by a mathematical program.
- Let $\mathcal{P}$ be the set of feasible solutions of the program and let $\mathcal{Q} \subset \mathcal{P}$.
- Performance of an algorithm

$$\frac{\text{algorithm's solution over } \mathcal{P}}{\text{dual solution over } \mathcal{Q}}$$

Generalized resource augmentation

Definition

- Consider an optimization problem that can be formalized by a mathematical program.
- Let $\mathcal{P}$ be the set of feasible solutions of the program and let $\mathcal{Q} \subset \mathcal{P}$.
- Performance of an algorithm

$$\frac{\text{algorithm's solution over } \mathcal{P}}{\text{dual solution over } \mathcal{Q}}$$

Examples

- speed-augmentation
  - constraint: *"each machine executes at most one job at each time"* $\Rightarrow$ *"the speed of the machine is one"*
  - the algorithm uses speed bigger than one (largest polytope)

Generalized resource augmentation

Definition

- Consider an optimization problem that can be formalized by a mathematical program.
- Let $\mathcal{P}$ be the set of feasible solutions of the program and let $\mathcal{Q} \subset \mathcal{P}$.
- Performance of an algorithm

$$\frac{\text{algorithm's solution over } \mathcal{P}}{\text{dual solution over } \mathcal{Q}}$$

Examples

- rejection
  - less jobs executed by the algorithm
    $\Rightarrow$ less constraints
    $\Rightarrow$ largest polytope

Generalized resource augmentation

Definition

- Consider an optimization problem that can be formalized by a mathematical program.
- Let $\mathcal{P}$ be the set of feasible solutions of the program and let $\mathcal{Q} \subset \mathcal{P}$.
- Performance of an algorithm

$$\frac{\text{algorithm's solution over } \mathcal{P}}{\text{dual solution over } \mathcal{Q}}$$

Questions:

1. Can we define other resource augmentation models ?
2. Can we apply this resource augmentation framework in other problems ?

Thank you !