Online scheduling to maximize throughput

Nguyen Kim Thang
(joint work with Christoph Durr and Lukasz Jez)

Online Algorithms

- No (partial) information about future (requests are revealed litle by litle).

- Guarantee the performance according to certain objective

Online Algorithms

- No (partial) information about future (requests are revealed litle by litle).

- Guarantee the performance according to certain objective

Competitive ratio

- An algorithm ALG is α-competitive if for any instance I

$$\frac{OPT(I)}{ALG(I)} \leq \alpha \quad \text{(maximization problem)}$$

Competitive ratio

- An algorithm ALG is α-competitive if for any instance I

$$\frac{OPT(I)}{ALG(I)} \leq \alpha \quad \text{(maximization problem)}$$

- What is a competitive ratio?
  - Measure the performance of an algorithm (worst-case analysis)
  - The price of an object (the problem):

$$\underset{\text{(upper bound)}}{\text{Algorithm}} \xleftrightarrow{\text{negociation}} \underset{\text{(lower bound)}}{\text{Adversary}}$$

  - An algorithm is optimal if the bound is tight.

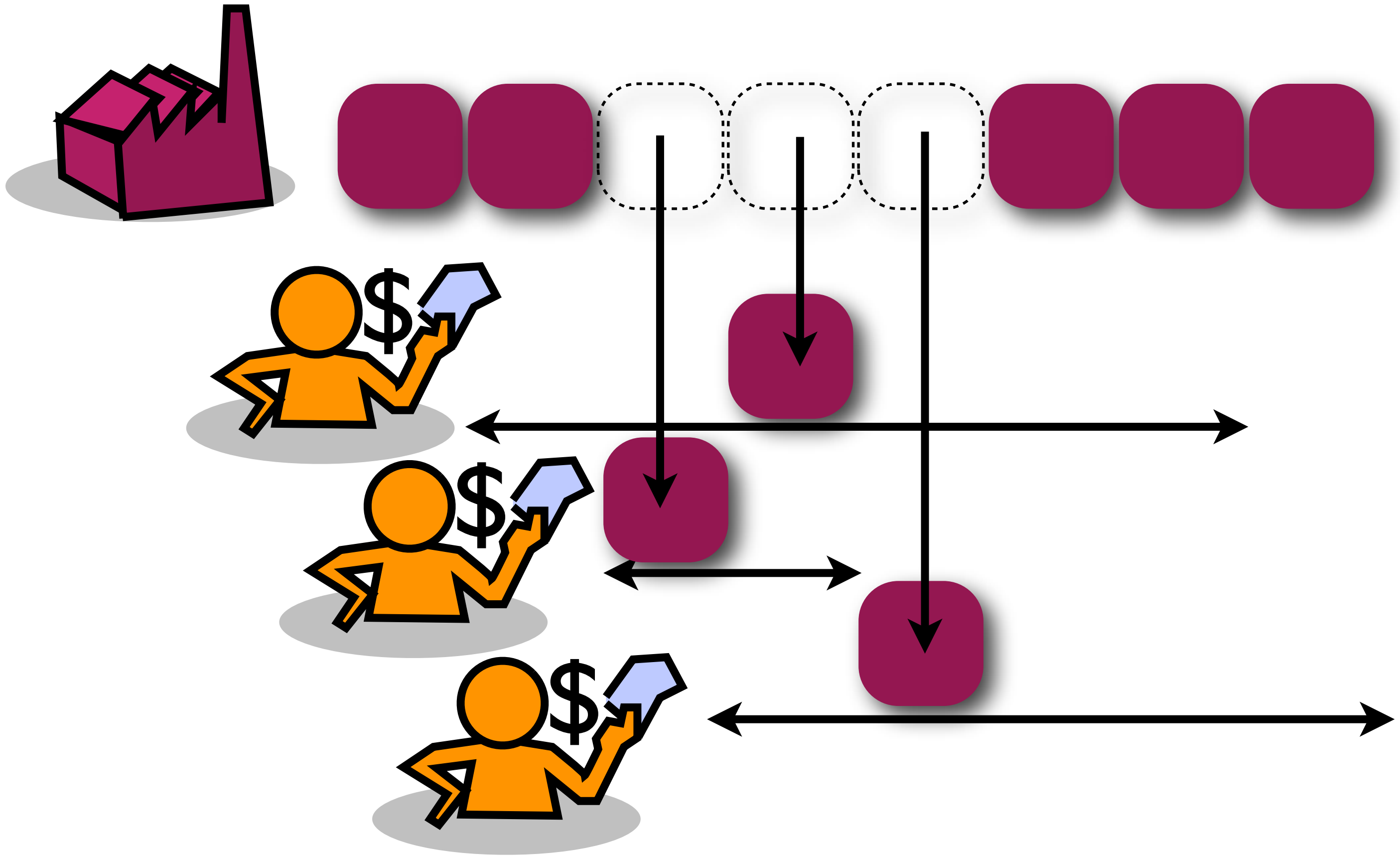
Model

Profit maximization

Enterprise: perishable product (electricity, ice-cream, ...).

Clients: arrive online, different demands.

Goal: maximize the profit.



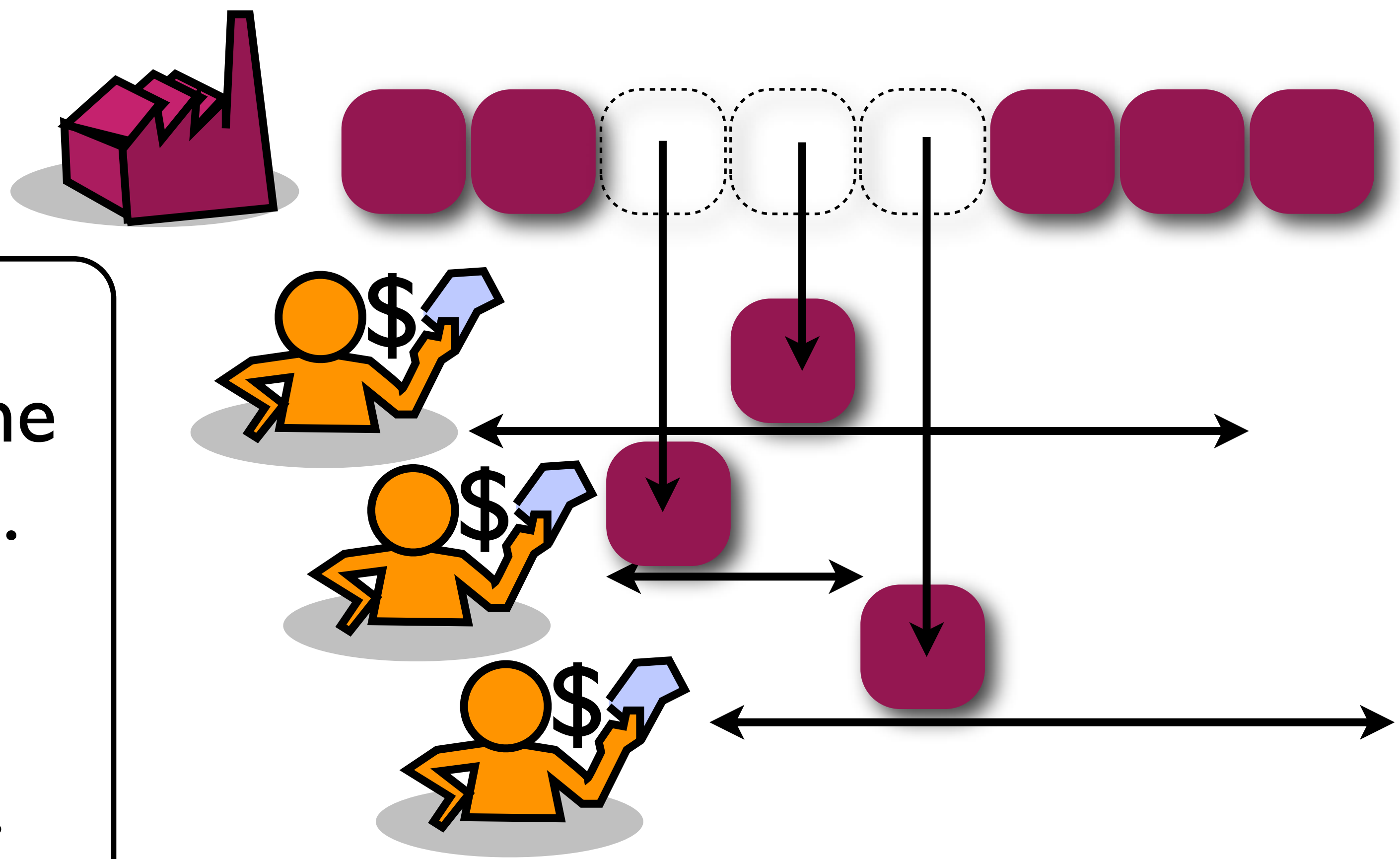
Model

Profit maximization

Enterprise: perishable product (electricity, ice-cream, ...).

Clients: arrive online, different demands.

Goal: maximize the profit.



Online Scheduling

Jobs: arrive at r_i, processing time p_i, deadline d_i, value (weight) w_i.

Objective: maximize the total value of jobs completed on time.

Contribution

	equal processing times	bounded processing times (by k)	unbounded processing times
unit weight	$\alpha = 1$	$\alpha = \Theta(\log k)$	∞
general	$\frac{3\sqrt{3}}{2} \leq \alpha \leq 5$ ≤ 4.24	$\alpha = \Theta(k/\log k)$	∞

Contribution

	equal processing times	bounded processing times (by k)	unbounded processing times
unit weight	$\alpha = 1$	$\alpha = \Theta(\log k)$	∞
general	$\frac{3\sqrt{3}}{2} \leq \alpha \leq 5$ ≤ 4.24	$\alpha = \Theta(k/\log k)$	∞

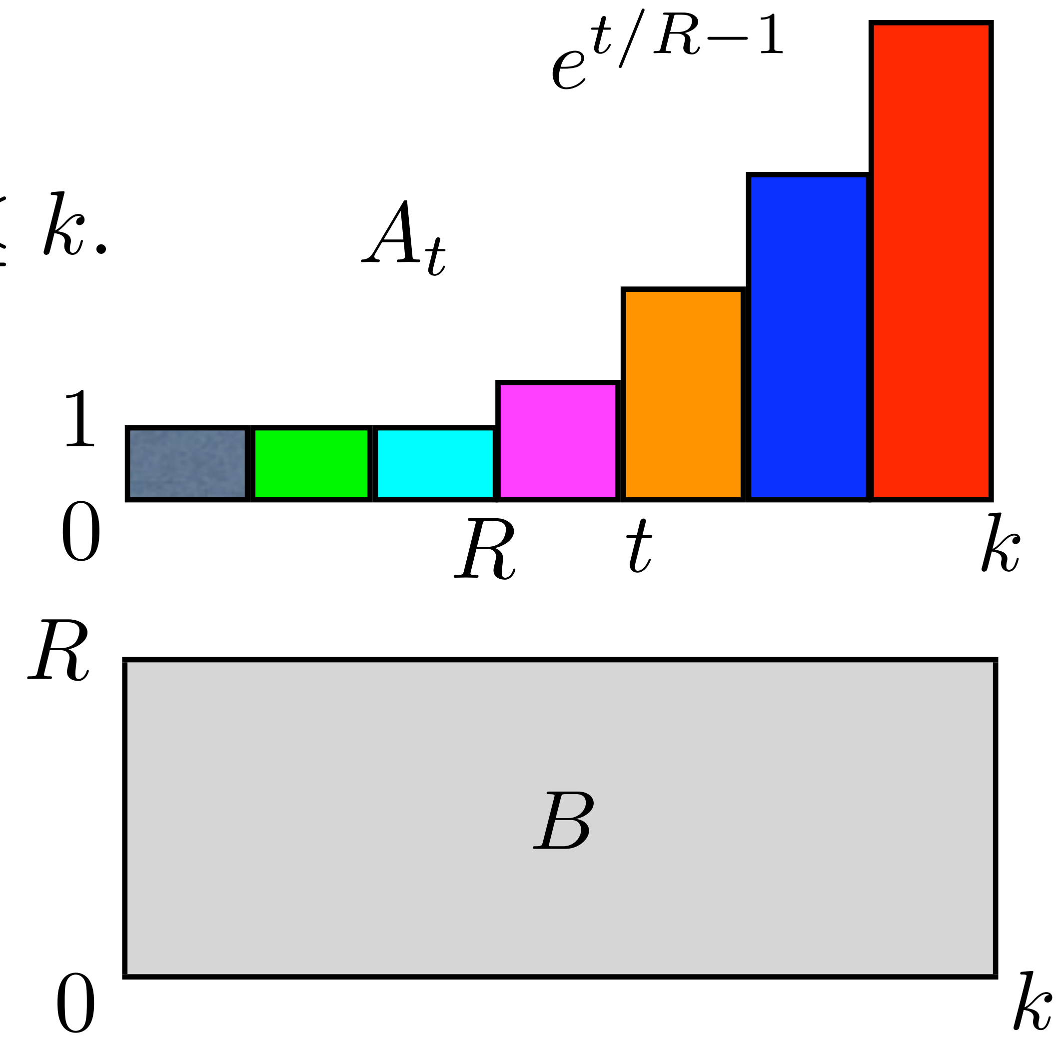
- ☑ **(Simple) Optimal algorithms (up to a constant)**
- ☑ **General analysis framework**

Lower bound (arbitrary weights)

- Theorem: any deterministic algorithm has competitive ratio $\Omega(k/\log k)$
- Proof:
  - Fix $R = k/\ln k$
  - Define:

$$A_t = \begin{cases} 1 \ \forall 0 \leq t < R \\ e^{\frac{t}{R}-1} \ \forall \ R \leq t \leq k. \end{cases}$$

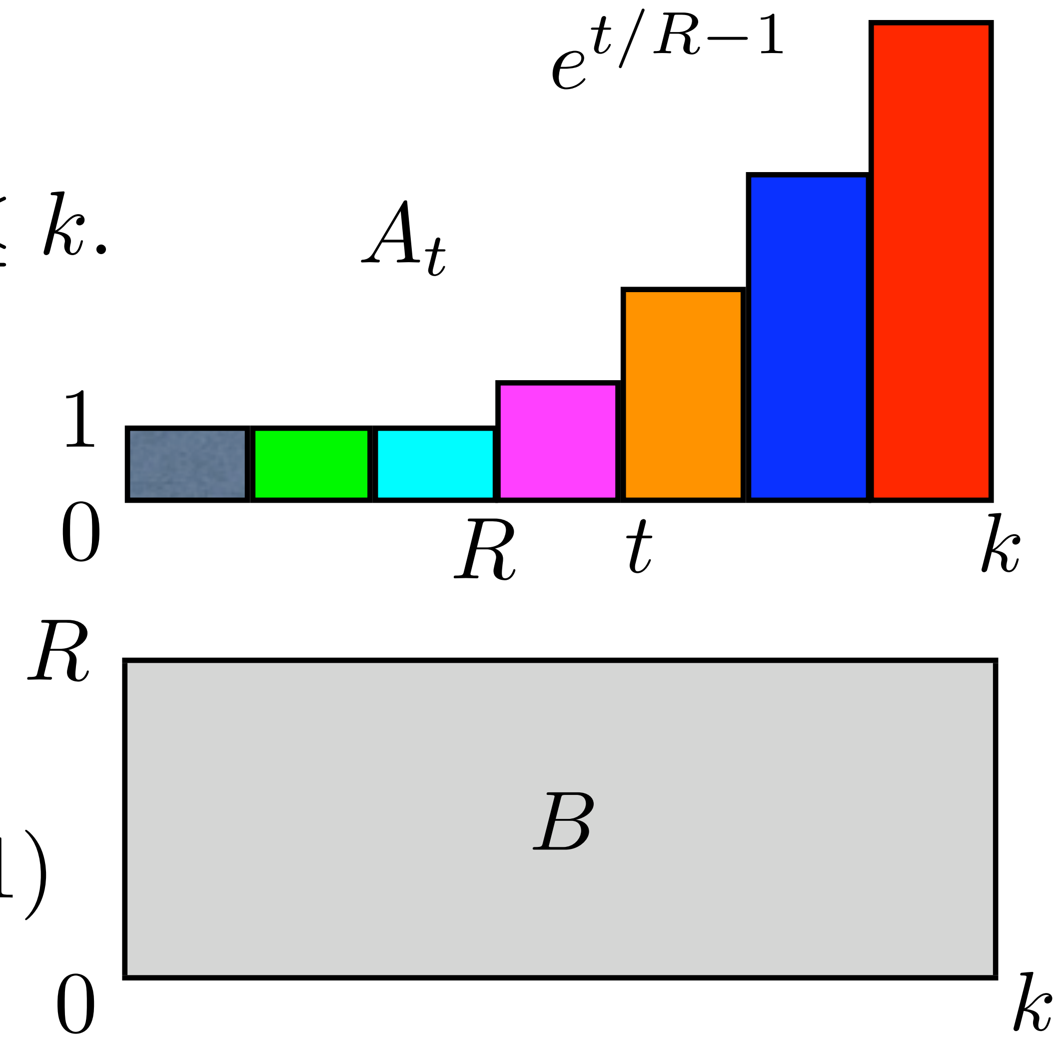
$$B = R$$



Lower bound (arbitrary weights)

- Theorem: any deterministic algorithm has competitive ratio
  $\Omega(k/\log k)$

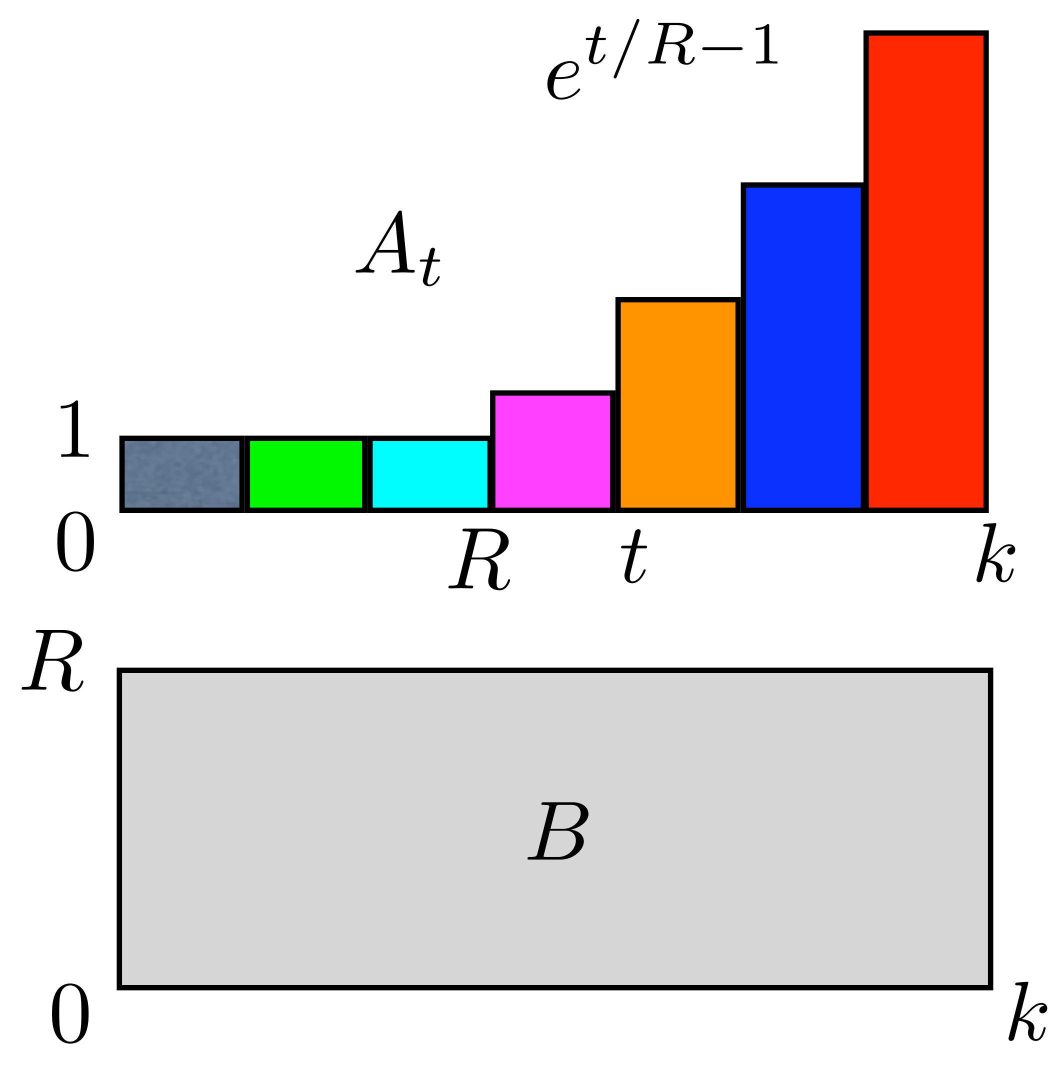
- Proof:
  - Fix $R = k/\ln k$
  - Define:
    $$A_t = \begin{cases} 1 \ \forall 0 \le t < R \\ e^{\frac{t}{R}-1} \ \forall \ R \le t \le k. \end{cases}$$
    $$B = R$$
  - Instance:
    - At time 0: release job B and A_0
    - At time t: release job A_t
      if ALG schedules B at time $(t-1)$
      otherwise, stop.
    - At time k: stop



Lower bound (arbitrary weights)

- If ALG schedules $A_{t^*} (0 \leq t^* < R)$ then ADV schedules B

  ratio = $R/1$



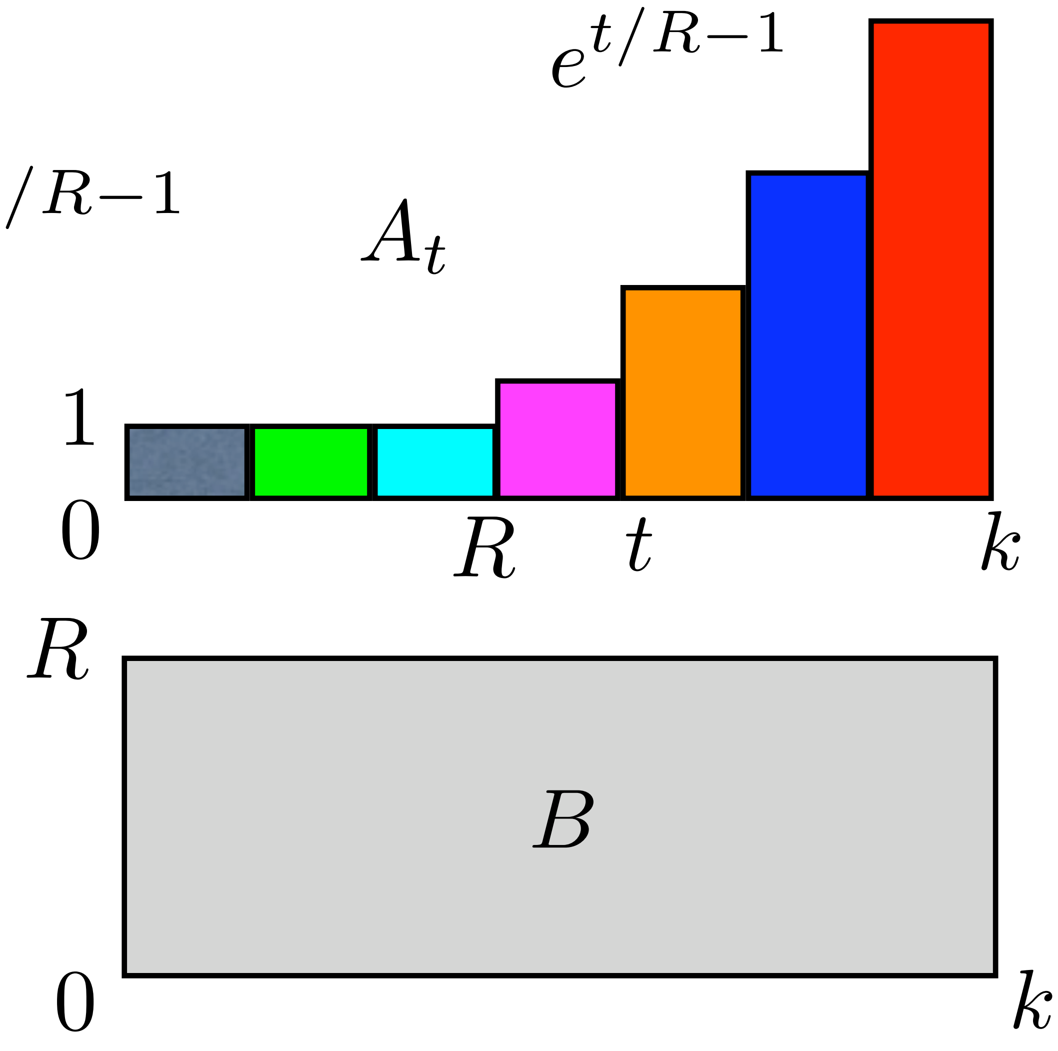
Lower bound (arbitrary weights)

- If ALG schedules $A_{t^*} (0 \leq t^* < R)$ then ADV schedules B

  ratio = $R/1$

- If ALG schedules $A_{t^*} (R \leq t^* \leq k)$ then ADV schedules all jobs $A_t (0 \leq t \leq t^*)$

$$\lceil R \rceil - 1 + \int_R^{t^*} e^{t/R-1} dt \geq R \cdot e^{t^*/R-1}$$



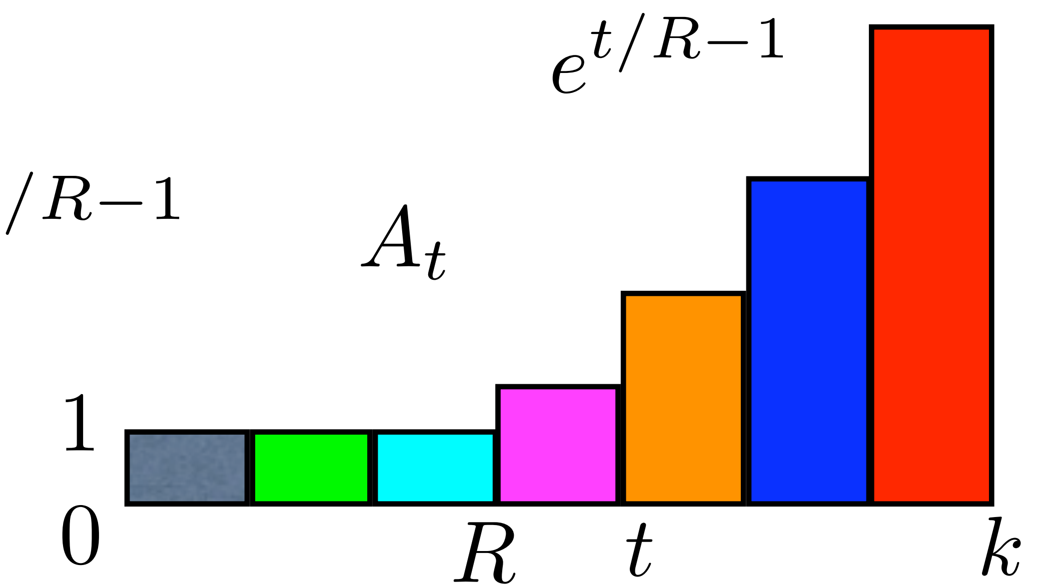
Lower bound (arbitrary weights)

- If ALG schedules $A_{t^*} (0 \leq t^* < R)$ then ADV schedules B

  ratio = $R/1$

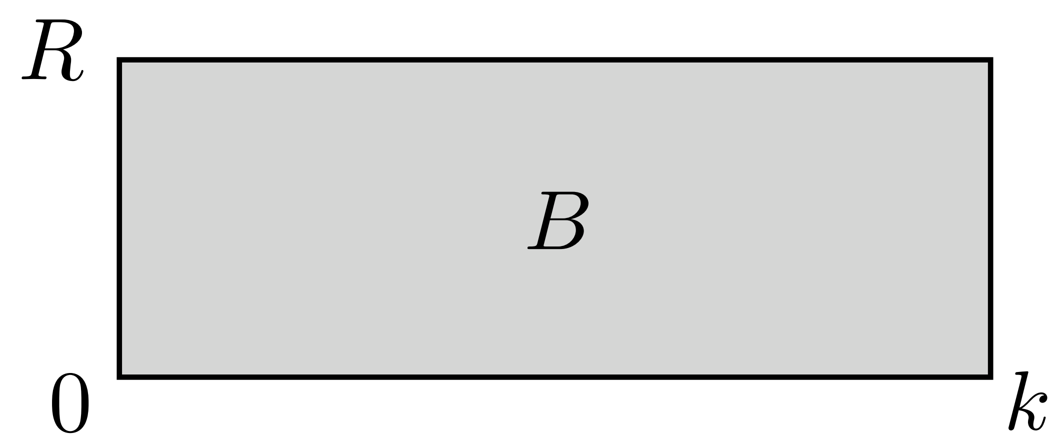
- If ALG schedules $A_{t^*} (R \leq t^* \leq k)$ then ADV schedules all jobs $A_t (0 \leq t \leq t^*)$

$$\lceil R \rceil - 1 + \int_R^{t^*} e^{t/R-1} dt \geq R \cdot e^{t^*/R-1}$$



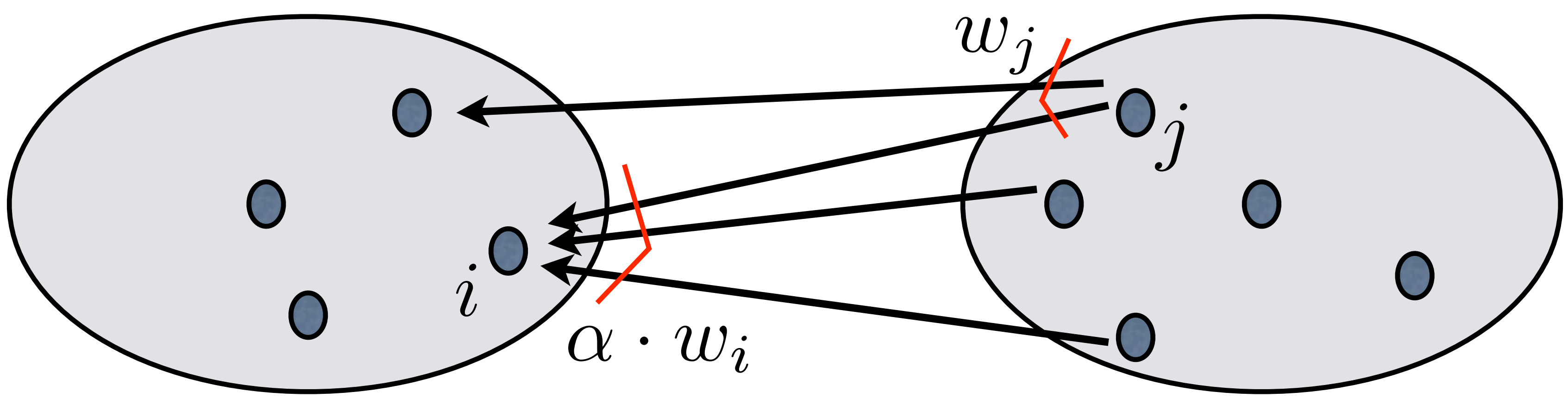
- If ALG completes B then ADV schedules all jobs $A_t (0 \leq t^* \leq k)$

$$Re^{k/R-1} = Rk/e \geq R^2$$



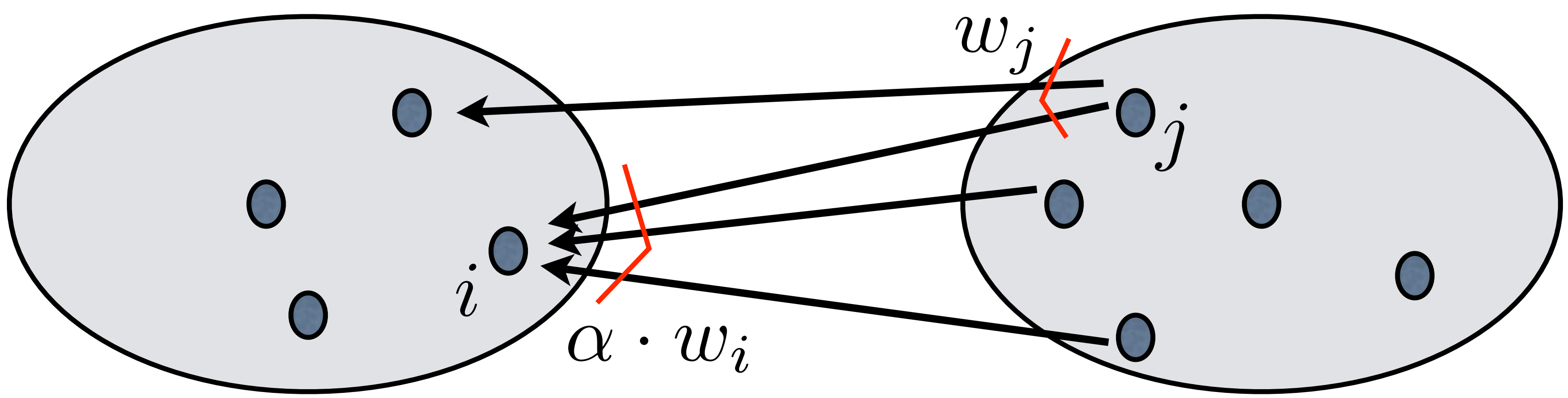
Settling the competitivity

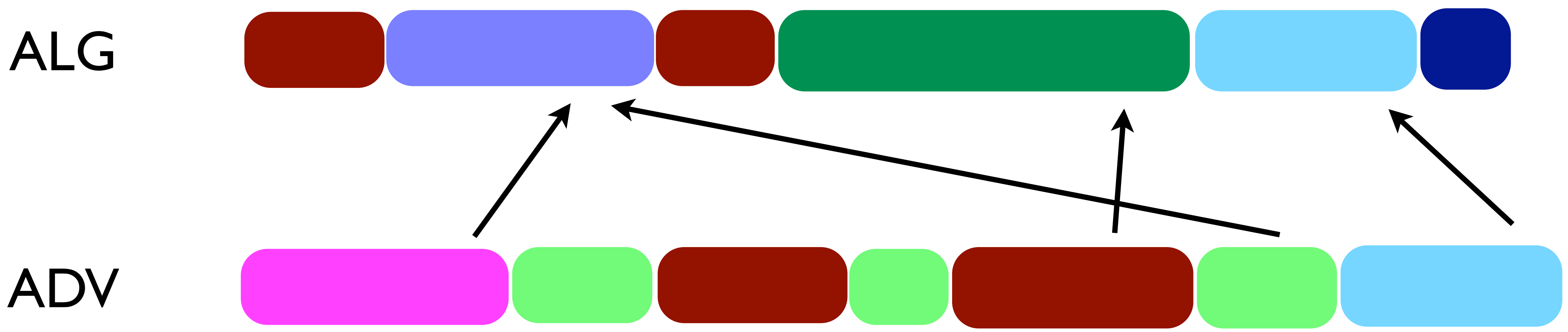
- Methods: charging scheme, potential function, etc



Settling the competitivity

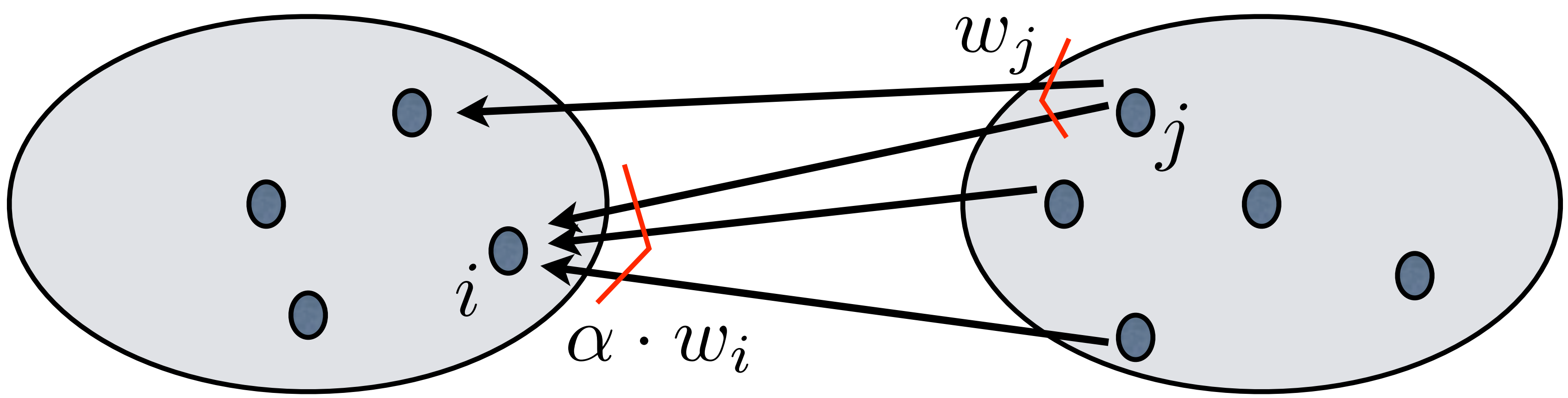
- Methods: charging scheme, potential function, etc

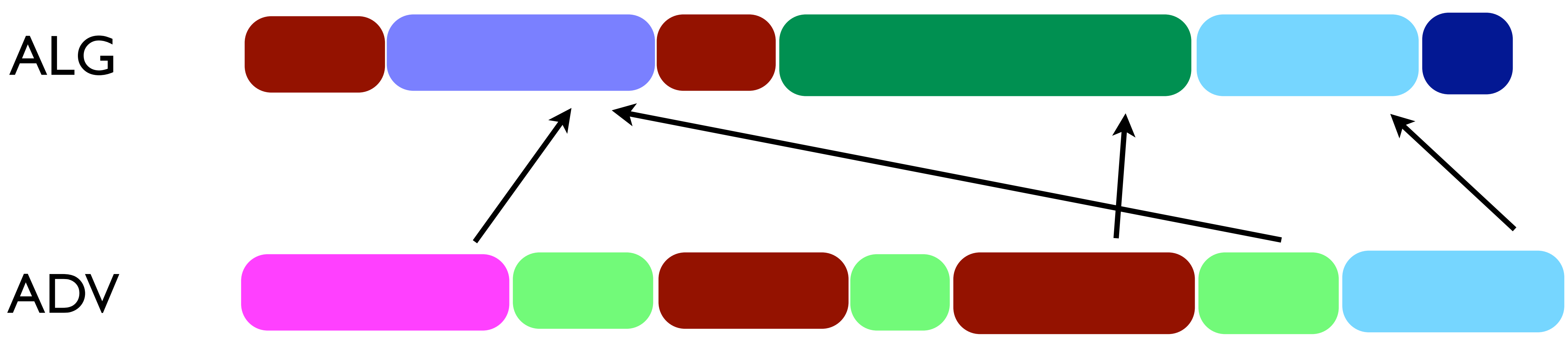




Settling the competitivity

- Methods: charging scheme, potential function, etc





- Def: An unit (i, a) is scheduled at time t if job i is scheduled at that time and the remaining processing time of i is a

Properties of algorithms

- A job is pending at time t if $t + q_j \leq d_j$
- A critical time of a job is the latest moment that the job is still pending.
- Associate (i, a) to a capacity function $\pi(i, a)$

Properties of algorithms

- A job is pending at time t if $t + q_j \leq d_j$
- A critical time of a job is the latest moment that the job is still pending.
- Associate (i, a) to a capacity function $\pi(i, a)$

- Desirable properties:
  - validity: if job j is pending for ALG at t then ALG schedules a unit (i, a) at t such that $\pi(i, a) \geq w_j / p_j$

Properties of algorithms

- A job is pending at time t if $t + q_j \leq d_j$
- A critical time of a job is the latest moment that the job is still pending.
- Associate (i, a) to a capacity function $\pi(i, a)$

- Desirable properties:
  - validity: if job j is pending for ALG at t then ALG schedules a unit (i, a) at t such that $\pi(i, a) \geq w_j / p_j$
  - ρ-monotonicity: if the ALG schedules (i, a) with $a > 1$ and (i', a') at $(t + 1)$ then $\rho \cdot \pi(i', a') \geq \pi(i, a)$

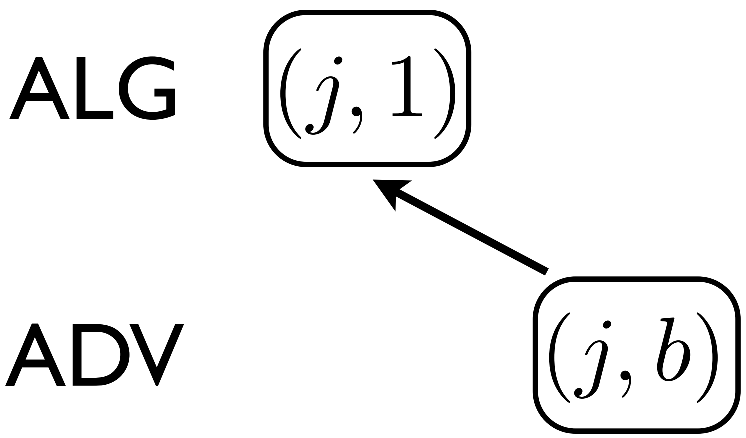
General charging scheme

- Each unit of job j completed by ADV has weight w_j/p_j

General charging scheme

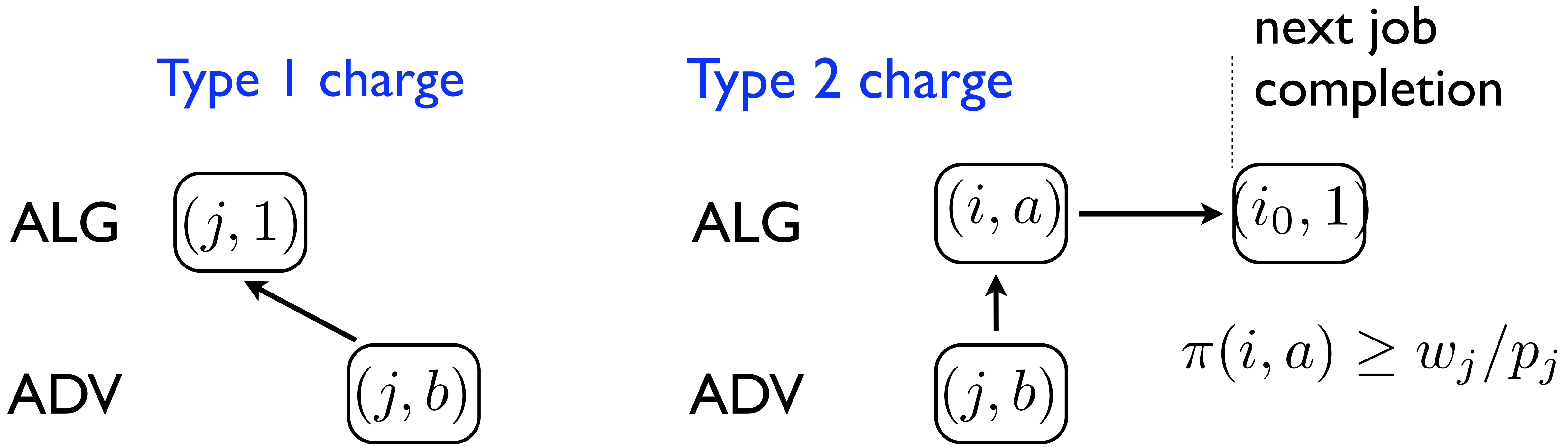
- Each unit of job j completed by ADV has weight w_j/p_j

Type 1 charge



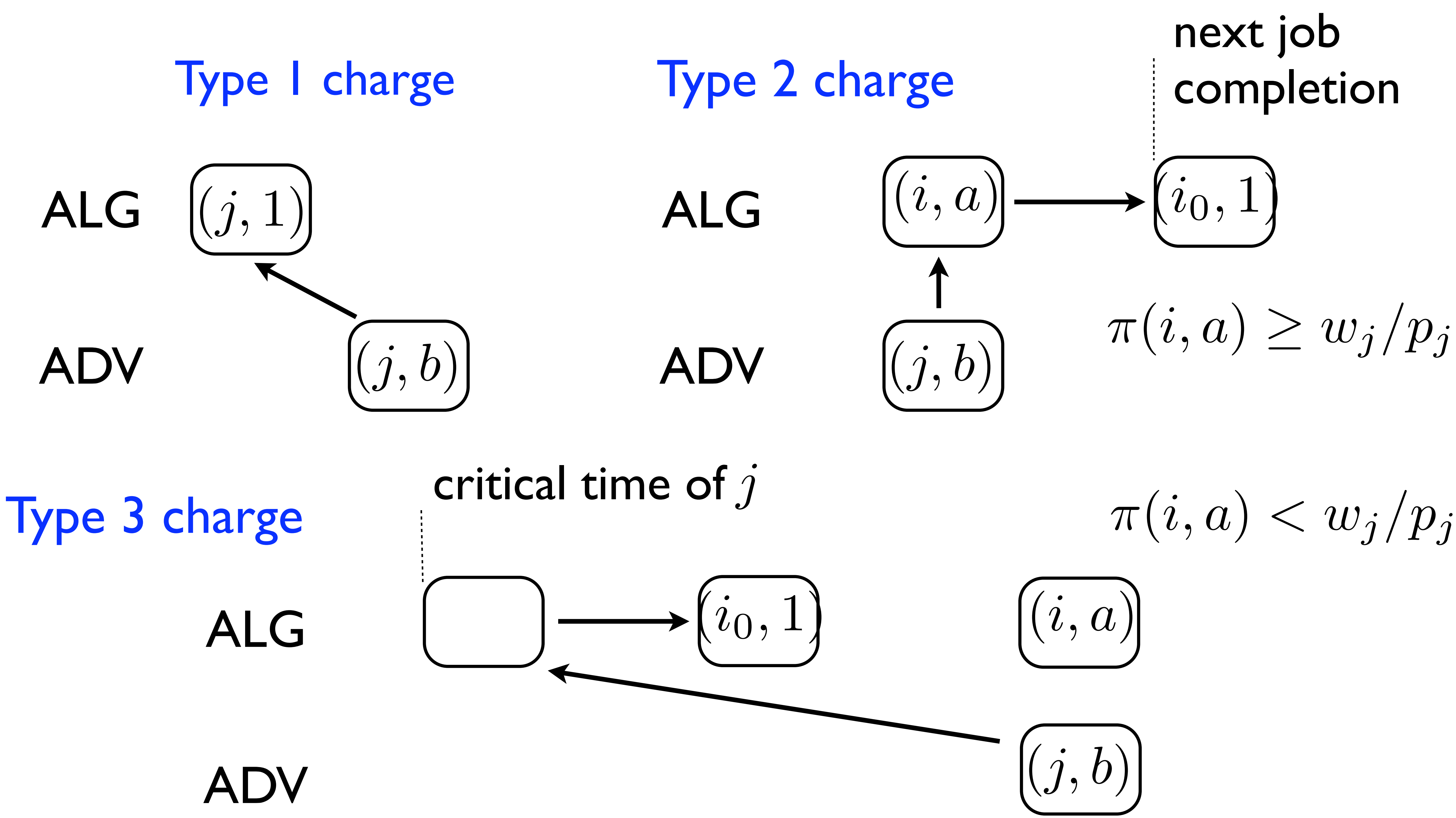
General charging scheme

- Each unit of job j completed by ADV has weight w_j/p_j



General charging scheme

- Each unit of job j completed by ADV has weight w_j/p_j

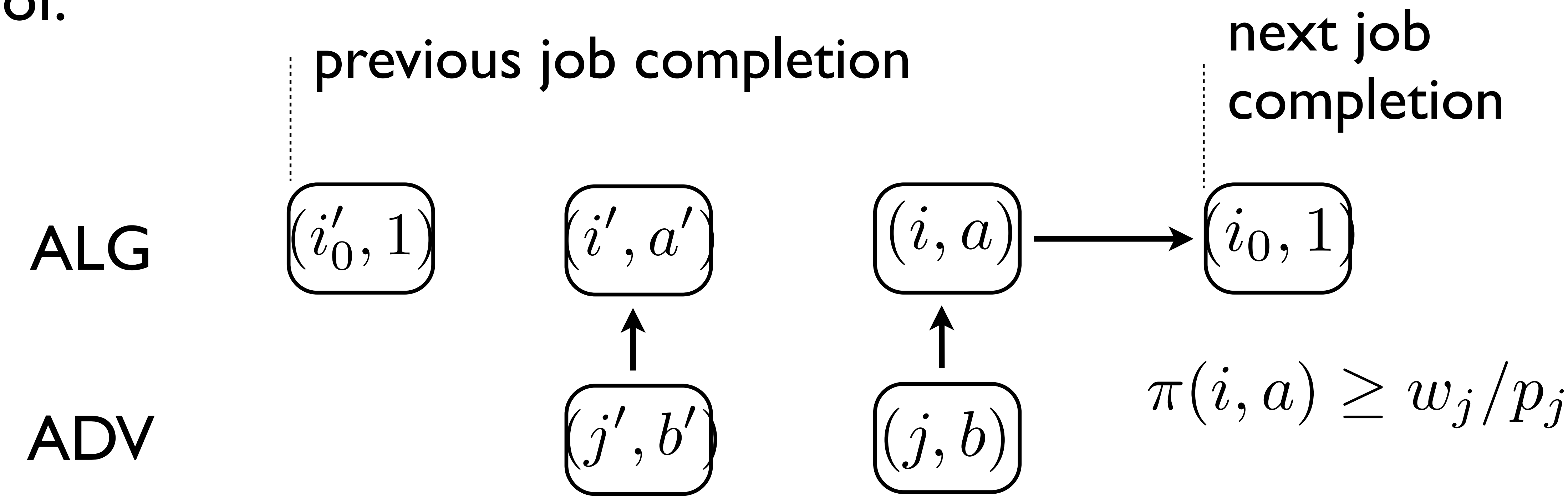


Main lemmas

- Lemma: a job i_0 receives at most $\dfrac{\pi(i_0, 1)}{1-\rho}$ charge of type 2

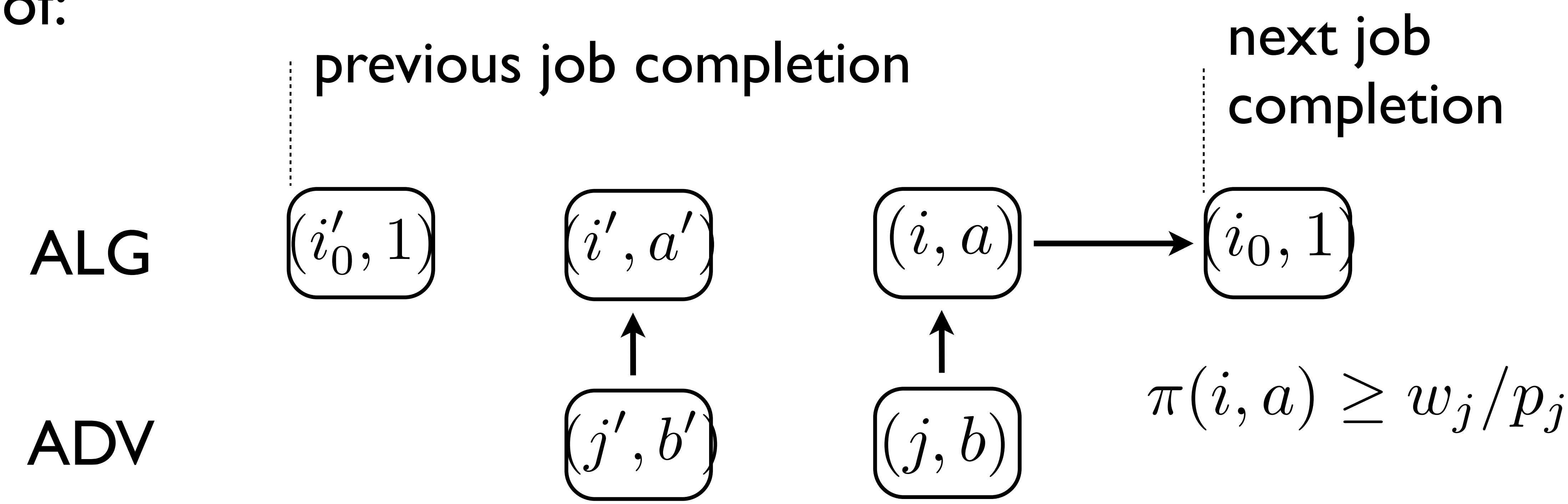
Main lemmas

- Lemma: a job i_0 receives at most $\dfrac{\pi(i_0, 1)}{1-\rho}$ charge of type 2
- Proof:



Main lemmas

- ☑ Lemma: a job i_0 receives at most $\dfrac{\pi(i_0, 1)}{1-\rho}$ charge of type 2
- ☐ Proof:



Total charge:

$$\sum_j \frac{w_j}{p_j} \leq \sum_i \pi(i,a) \leq \sum_k \rho^k \pi(i_0, 1) \leq \frac{\pi(i_0,1)}{1-\rho}$$

type 2 monotonicity

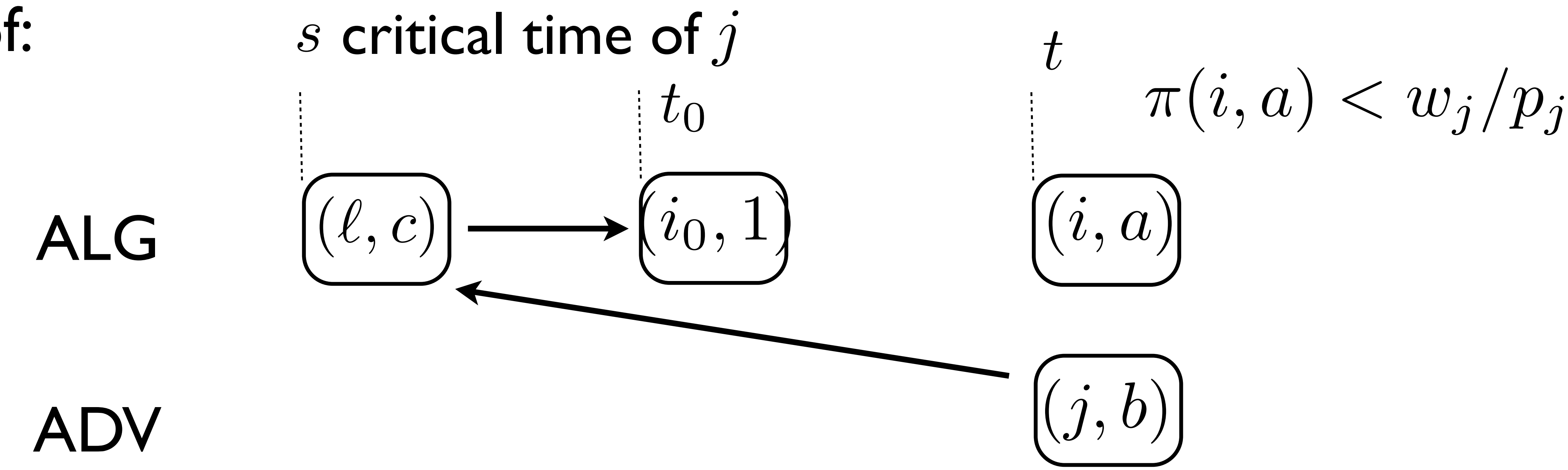
Main lemmas

- Lemma: J set of job units that are type 3 charged to i_0

  Then $|J| \leq k-1$ and $w_j/p_j \leq \pi(i_0, 1)\ \forall j \in J$

Main lemmas

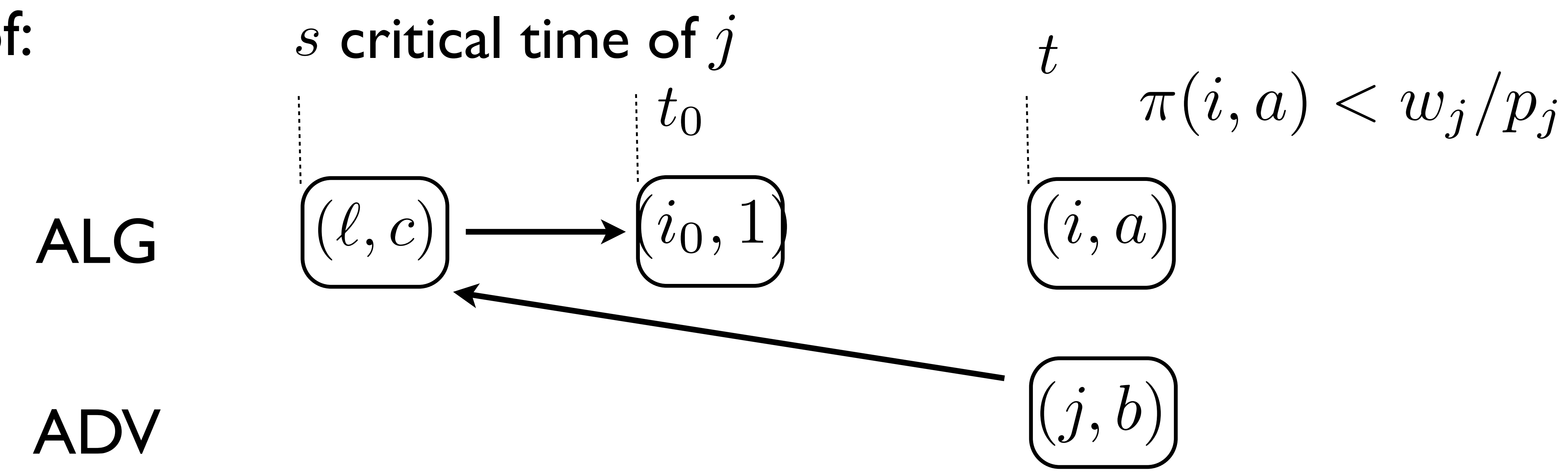
- Lemma: J set of job units that are type 3 charged to i_0
  Then $|J| \leq k-1$ and $w_j/p_j \leq \pi(i_0,1)\ \forall j \in J$
- Proof:



Main lemmas

- Lemma: J set of job units that are type 3 charged to i_0

  Then $|J| \leq k-1$ and $w_j/p_j \leq \pi(i_0,1)\ \forall j \in J$
- Proof:



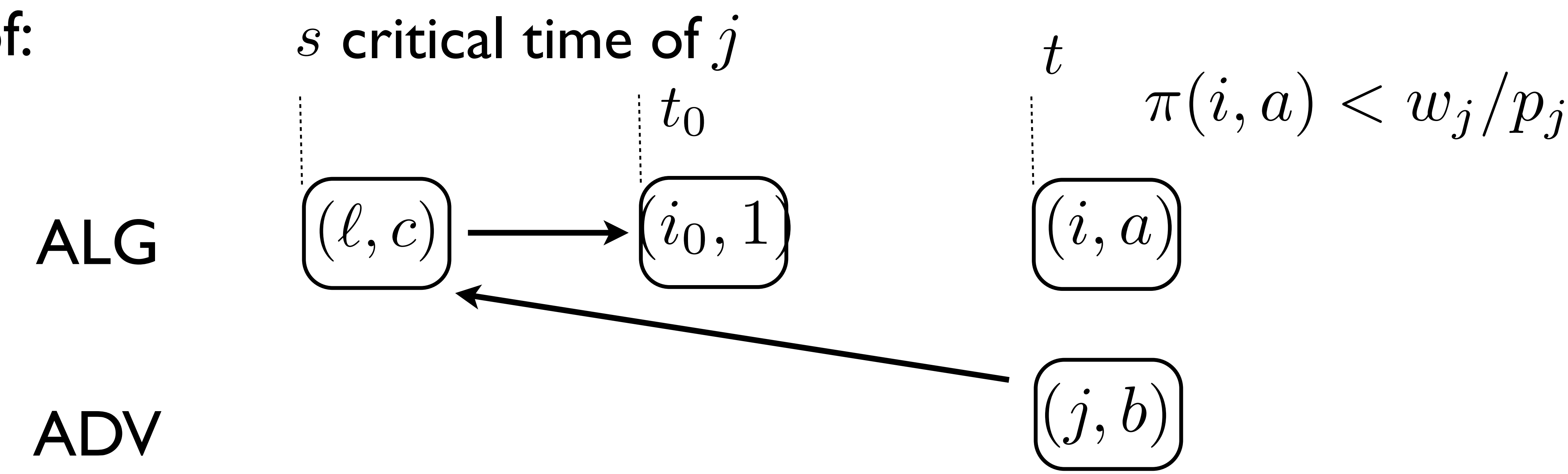
  - $t_0 \leq t$: otherwise, $w_j/p_j \leq \pi(\ell,c) \leq \pi(i,a)$

Main lemmas

- Lemma: J set of job units that are type 3 charged to i_0

  Then $|J| \leq k-1$ and $w_j/p_j \leq \pi(i_0, 1)\ \forall j \in J$

- Proof:



  - $t_0 \leq t$: otherwise, $w_j/p_j \leq \pi(\ell, c) \leq \pi(i, a)$

  - s is critical time $s + q_j(s) = d_j$

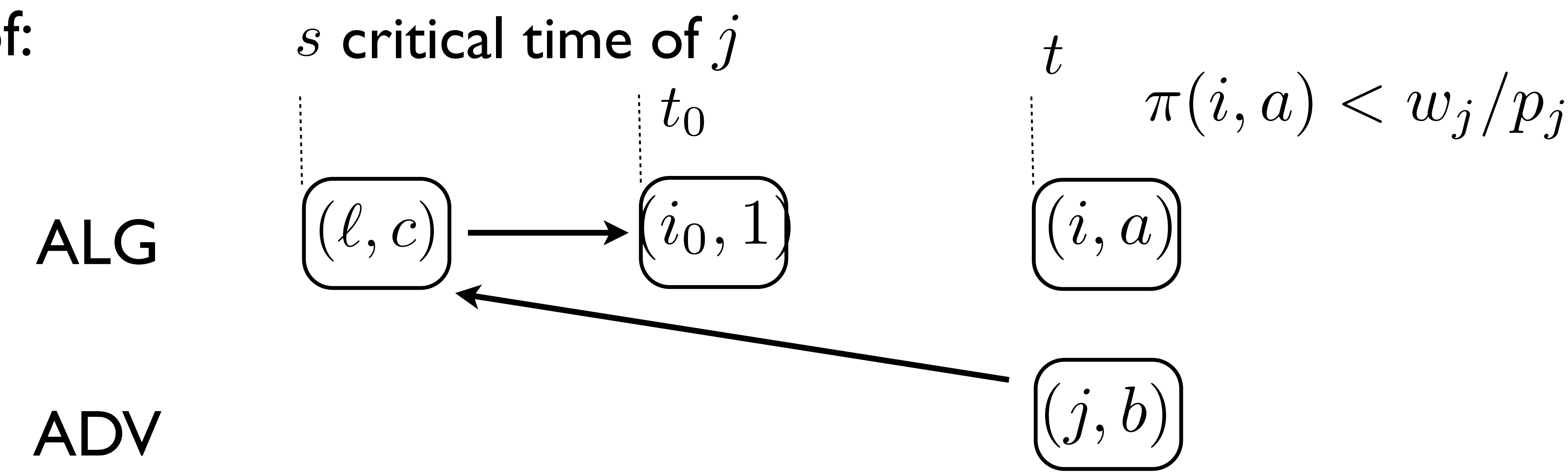
    Moreover $t < d_j$

Main lemmas

- Lemma: J set of job units that are type 3 charged to i_0

  Then $|J| \leq k-1$ and $w_j/p_j \leq \pi(i_0, 1)\ \forall j \in J$

- Proof:



  - $t_0 \leq t$: otherwise, $w_j/p_j \leq \pi(\ell, c) \leq \pi(i, a)$
  - s is critical time $s + q_j(s) = d_j$ Then:

    Moreover $t < d_j$

    $$t - s < q_j(s) \leq p_j \leq k$$

Hence, $|J| \leq k-1$

Smith ratio algorithm

- Algorithm: at any time, schedule the pending job that maximizes w_j/p_j

Smith ratio algorithm

- Algorithm: at any time, schedule the pending job that maximizes w_j/p_j
- Theorem: the Smith ratio algorithm is $2k$-competitive

Smith ratio algorithm

- Algorithm: at any time, schedule the pending job that maximizes w_j/p_j
- Theorem: the Smith ratio algorithm is $2k$-competitive
- Proof: Define: $\pi(i,a) = w_i/a$
  - The algorithm satisfies validity: $\pi(i,a) \geq w_j/p_j$
  - The algorithm is $(k-1)/k$-monotone

Smith ratio algorithm

- **Algorithm:** at any time, schedule the pending job that maximizes w_j/p_j
- **Theorem:** the Smith ratio algorithm is $2k$-competitive
- **Proof:** Define: $\pi(i,a) = w_i/a$
  - The algorithm satisfies validity: $\pi(i,a) \geq w_j/p_j$
  - The algorithm is $(k-1)/k$-monotone

  Let (j,b) be an unit scheduled before (i,a)

If $(i,a) = (j,b-1)$ then $\frac{k-1}{k}\pi(i,a) = \frac{k-1}{k}\frac{w_j}{b-1} \geq \frac{w_j}{b} = \pi(j,b)$

Otherwise,

$$\frac{k-1}{k}\pi(i,a) = \frac{k-1}{k}\frac{w_i}{p_i} \geq \frac{k-1}{k}\frac{w_j}{p_j} \geq \frac{w_j}{b} = \pi(j,b)$$

Smith ratio algorithm

- **Algorithm:** at any time, schedule the pending job that maximizes w_j/p_j
- **Theorem:** the Smith ratio algorithm is $2k$-competitive
- **Proof:** Define: $\pi(i,a) = w_i/a$
  - The algorithm satisfies validity: $\pi(i,a) \geq w_j/p_j$
  - The algorithm is $(k-1)/k$-monotone

    Let (j,b) be an unit scheduled before (i,a)

If $(i,a) = (j,b-1)$ then $\frac{k-1}{k}\pi(i,a) = \frac{k-1}{k}\frac{w_j}{b-1} \geq \frac{w_j}{b} = \pi(j,b)$

Otherwise,

$$\frac{k-1}{k}\pi(i,a) = \frac{k-1}{k}\frac{w_i}{p_i} \geq \frac{k-1}{k}\frac{w_j}{p_j} \geq \frac{w_j}{b} = \pi(j,b)$$

Total charge of job i_0: $\quad w_{i_0} + \frac{w_{i_0}}{1-(k-1)/k} + (k-1)w_{i_0}$

Optimal algorithm

- **Algorithm:** at any time, schedule the pending job that maximizes

$$\pi(j, q_j) := w_j \alpha^{q_j - 1} := w_j \left(1 - c^2 \frac{\ln k}{k}\right)^{q_j - 1}$$

- **Theorem:** the Smith ratio algorithm is $(4k/\ln k)$-competitive

Optimal algorithm

- **Algorithm:** at any time, schedule the pending job that maximizes

$$\pi(j, q_j) := w_j \alpha^{q_j - 1} := w_j \left(1 - c^2 \frac{\ln k}{k}\right)^{q_j - 1}$$

- **Theorem:** the Smith ratio algorithm is $(4k/\ln k)$-competitive

- **Proof (sketch):**
  - The algorithm satisfies validity: $\pi(i, a) \geq w_j / p_j$
  - The algorithm is α-monotone

Total charge of job i_0: $\quad w_{i_0} + O\left(\frac{k}{\ln k}\right) w_{i_0} + \sum_{j \in J} \frac{w_{i_0}}{f(p_j)}$

where $f(x) = x\alpha^{x-1}$

Conclusion

☑ Optimal algorithms and a general framework for the model

☑ Constant competitive algorithms for a variant where jobs have equal lengths $\longrightarrow$ $O(\log k)$ competitive randomized algos.

☐ Directions: Close the gap

- Equal length jobs (2.59 < ratio < 4.25)
- Randomized algorithms $\Omega(\sqrt{\log k / \log\log k}), O(\log k)$

Thank you!