

Online Scheduling

Susanne Albers
University of Freiburg
Germany

Motivation

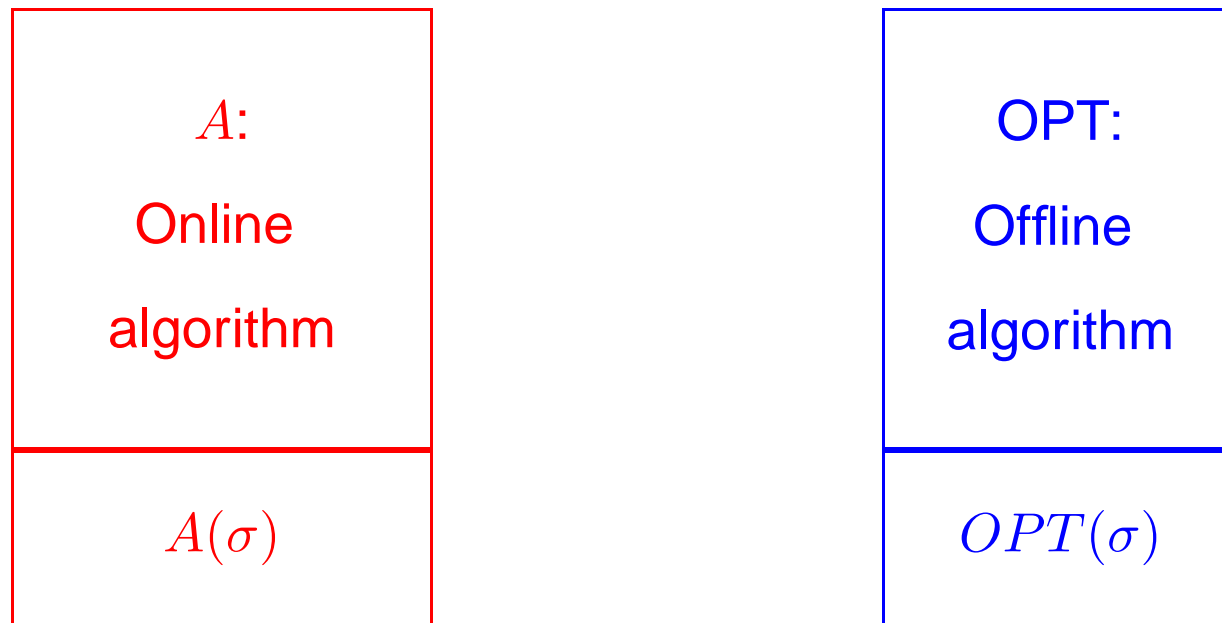
Decision making with incomplete information.

- Jobs arrive as a **sequence**, maybe even over time. Future jobs unknown.
- **Processing times** are unknown.
- Processor **breakdowns** or maintenance intervals are unknown.

Compute schedules that represent good approximate solutions.

Competitive analysis

Online setting: jobs sequence



A is c -competitive if for all job sequences σ

$$A(\sigma) \leq c \cdot OPT(\sigma).$$

Problems settings

m processors/machines

n jobs J_1, \dots, J_n p_{ij} = processing time J_i on machine j

- Machine environments: identical: $p_{ij} = p_i$
related: $p_{ij} = p_i/s_j$ for some processing time p_i and machine speed s_j
unrelated: p_{ij} may take arbitrary values
- Online models: jobs arrive as a sequence/list
jobs arrive over time
- Job duration: permanent or temporary.

- Objective functions:

Makespan C_{\max} sum of completion times $\sum_i c_i$ $\sum_i w_i c_i$

Flow time $f_i = c_i - a_i$ $\sum_i f_i$ $\sum_i w_i f_i$

Stretch $s_i = f_i/p_i$ $\sum_i s_i$ $\sum_i w_i s_i$

Graham's problem

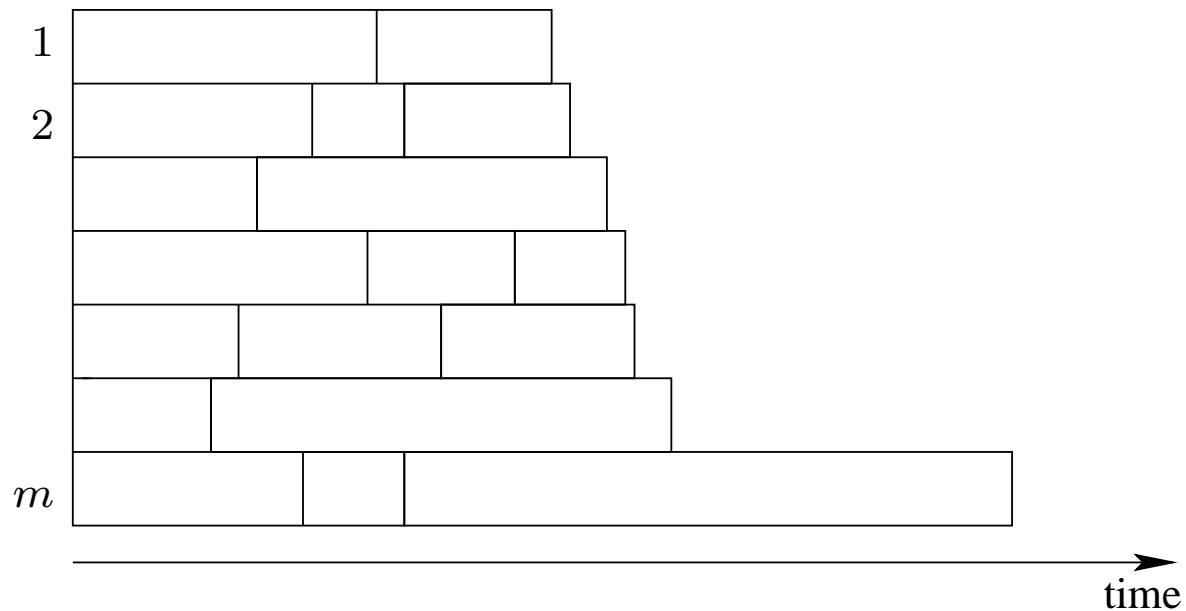
Graham 1966

m identical parallel machines.

Jobs J_1, J_2, \dots, J_n with processing times p_1, p_2, \dots, p_n .

Jobs arrive one by one. Preemption not allowed.

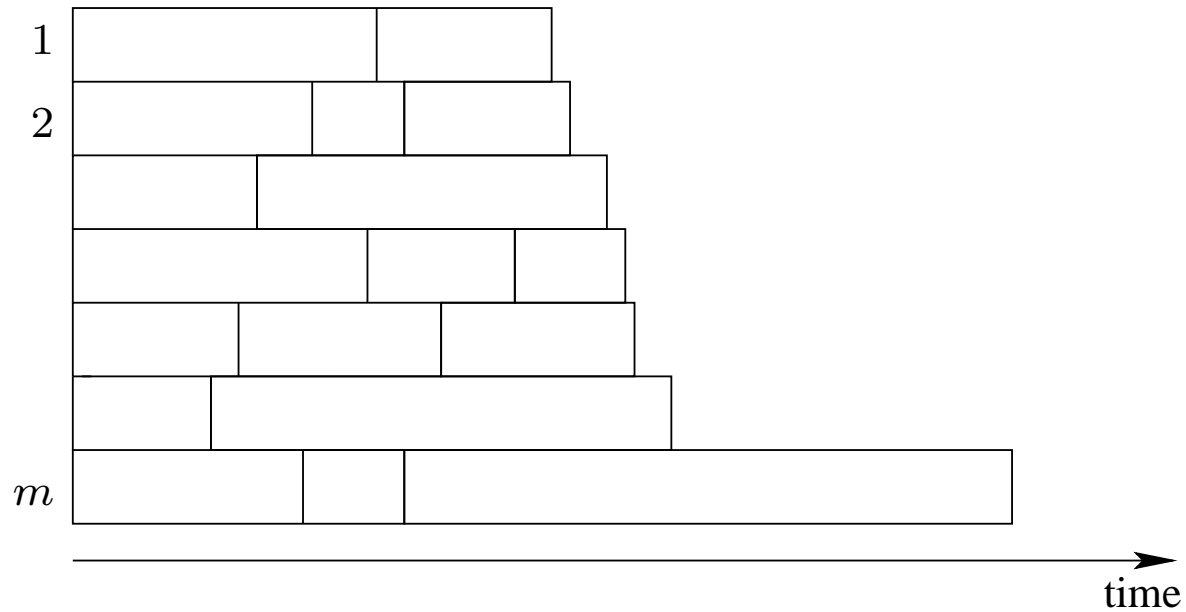
Minimize **makespan**.



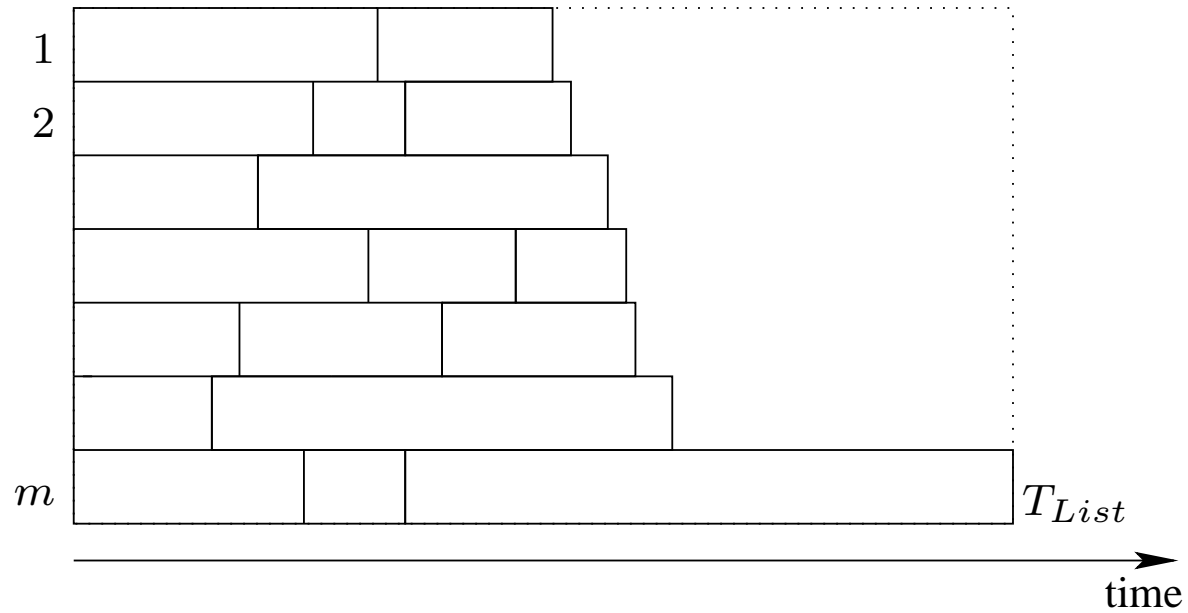
Graham's problem

List: Schedule any new job on least loaded machine.

Thm: List is $(2 - \frac{1}{m})$ -competitive.

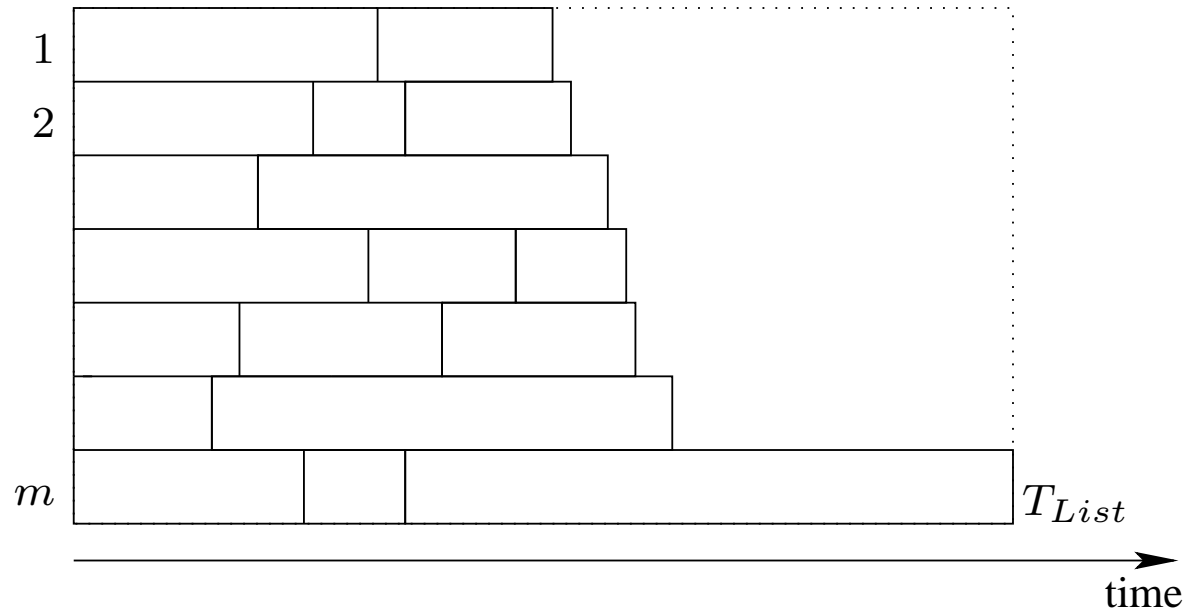


Analysis



$$mT_{List} \leq \sum_{i=1}^n p_i + (m-1) \max_{1 \leq i \leq n} p_i$$

Analysis



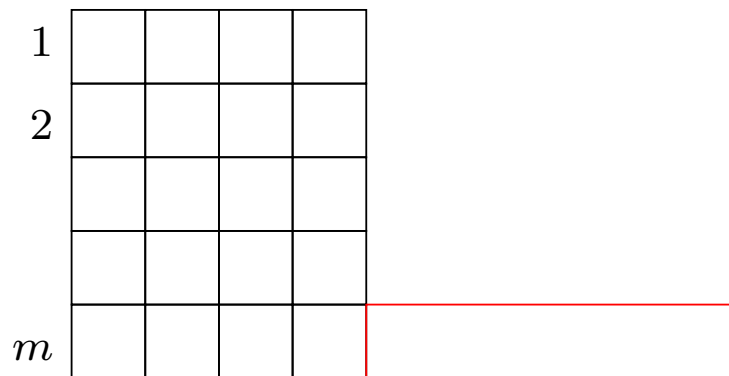
$$mT_{List} \leq \sum_{i=1}^n p_i + (m-1) \max_{1 \leq i \leq n} p_i$$

$$T_{List} \leq \frac{1}{m} \sum_{i=1}^n p_i + \left(1 - \frac{1}{m}\right) \max_{1 \leq i \leq n} p_i \leq T_{OPT} + \left(1 - \frac{1}{m}\right) T_{OPT}$$

Lower bound

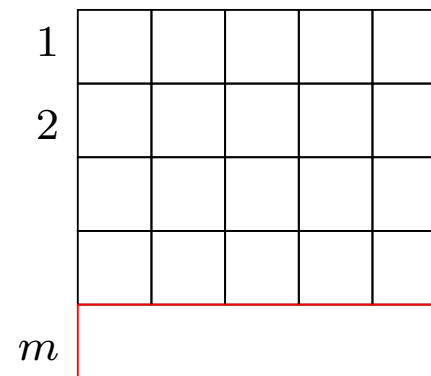
Thm: List is not better than $(2 - \frac{1}{m})$ -competitive.

$m(m - 1)$ jobs of size 1 1 job of size m



List

$$T_{List} = (m - 1) + m = 2m - 1$$



OPT

$$T_{List} = m$$

Improvements

List best possible for $m = 2, 3$

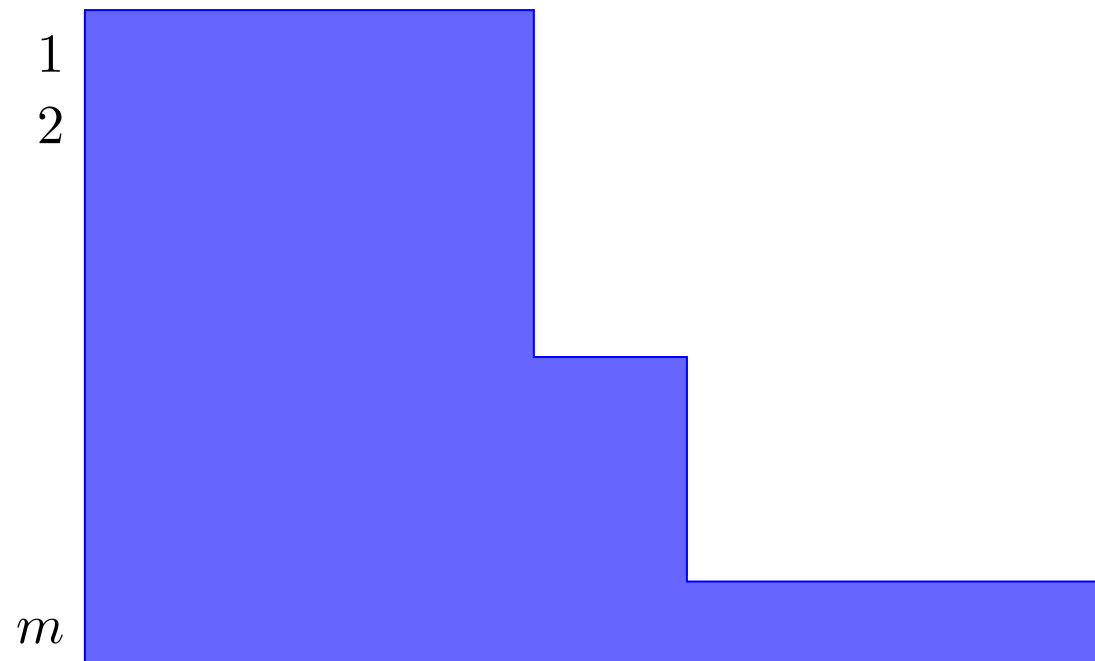
Upper bounds

- 1.986 Bartal, Fiat, Karloff, Vohra STOC 92
- 1.945 Karger, Phillips, Torng SODA 94
- 1.923 Albers STOC 97
- 1.9201 Fleischer, Wahl ESA 00

Lower bounds

- 1.701 Faige, Kern, Turan 89
- 1.837 Bartal, Karloff, Rabani 94
- 1.852 Albers 97
- 1.853 Gormey, Reingold, Torng, Westbrook 00
- 1.88 Rudin 01

Imbalanced schedules



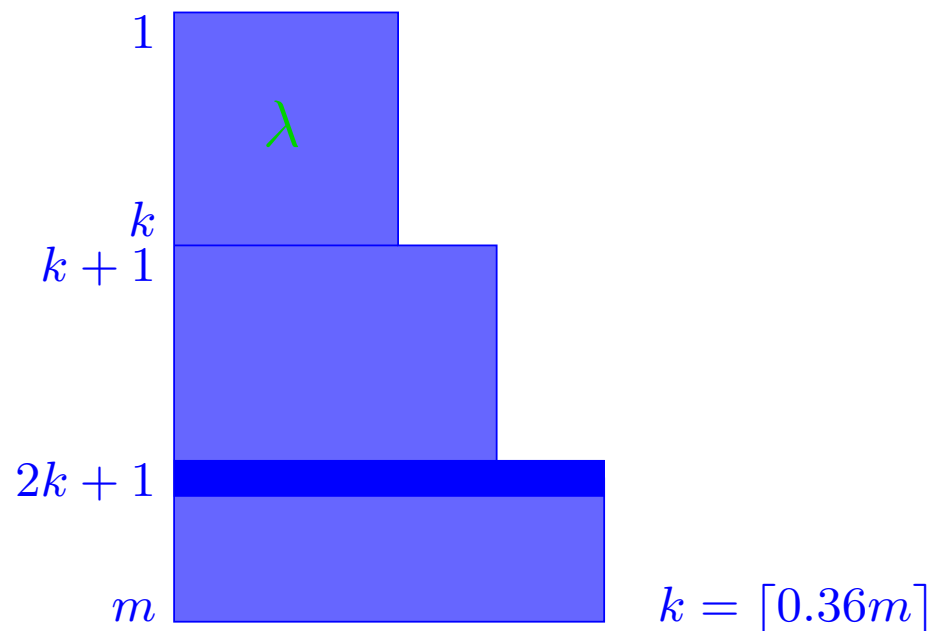
Makespan higher than necessary at this point in time

Algorithm Imbal

Sort machines in order of non-decreasing load

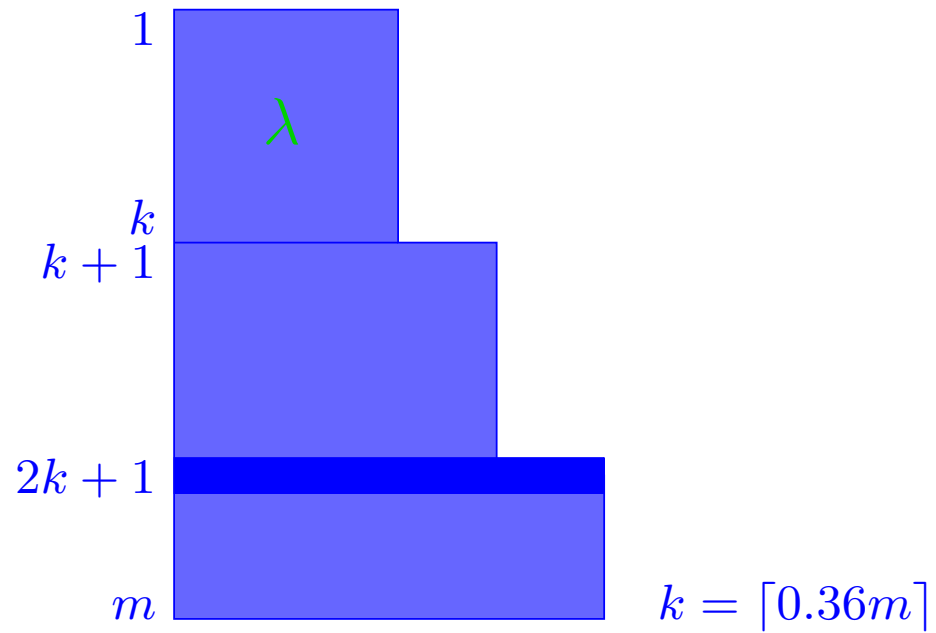
Load = processing times of jobs assigned to it

l_i = load of i -th smallest machine



$$\lambda = \frac{1}{k} \sum_{i=1}^k l_i \quad \text{average load on the } k \text{ smallest machines}$$

Algorithm Imbal

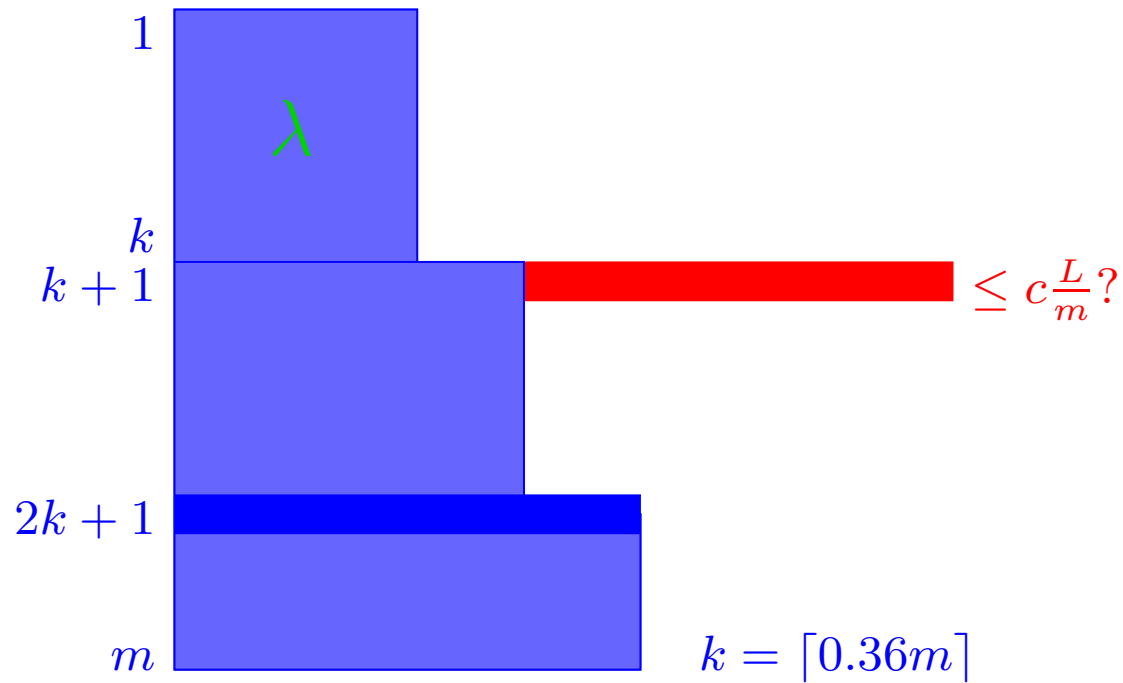


Try to maintain

$$\lambda \leq \alpha \cdot l_{2k+1}$$

$$\alpha = 0.456$$

Algorithm Imbal



Algorithm: $c = 1.9201$ $L = \sum_i l_i$

Schedule new job on machine $k+1$ if

- $\lambda > \alpha \cdot l_{2k+1}$ and resulting load $\leq c \cdot \frac{L}{m}$

Otherwise schedule job on machine 1.

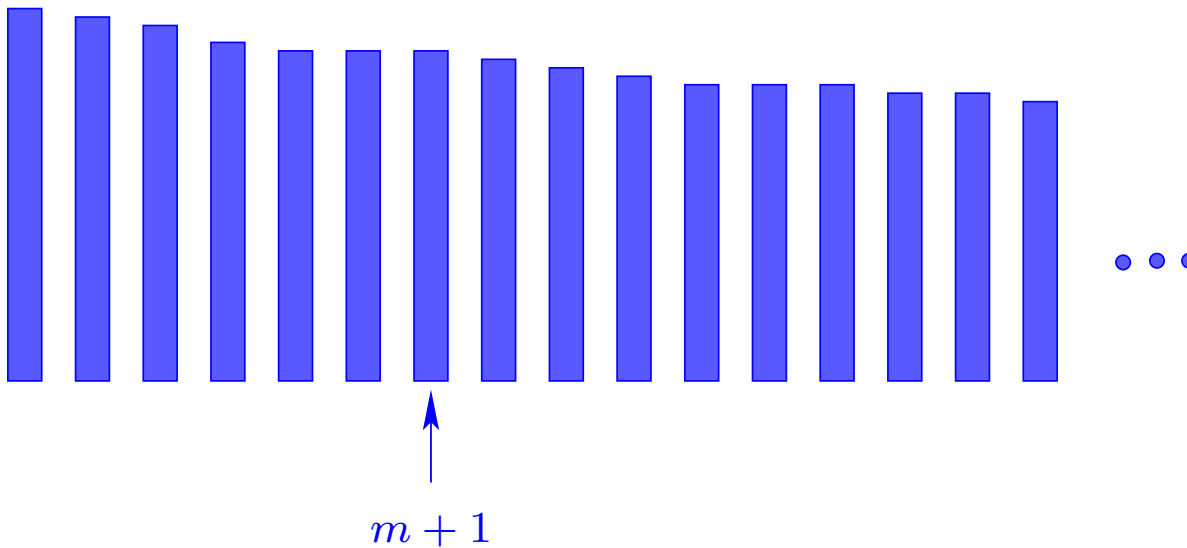
Room for improvement?

OPT is at least

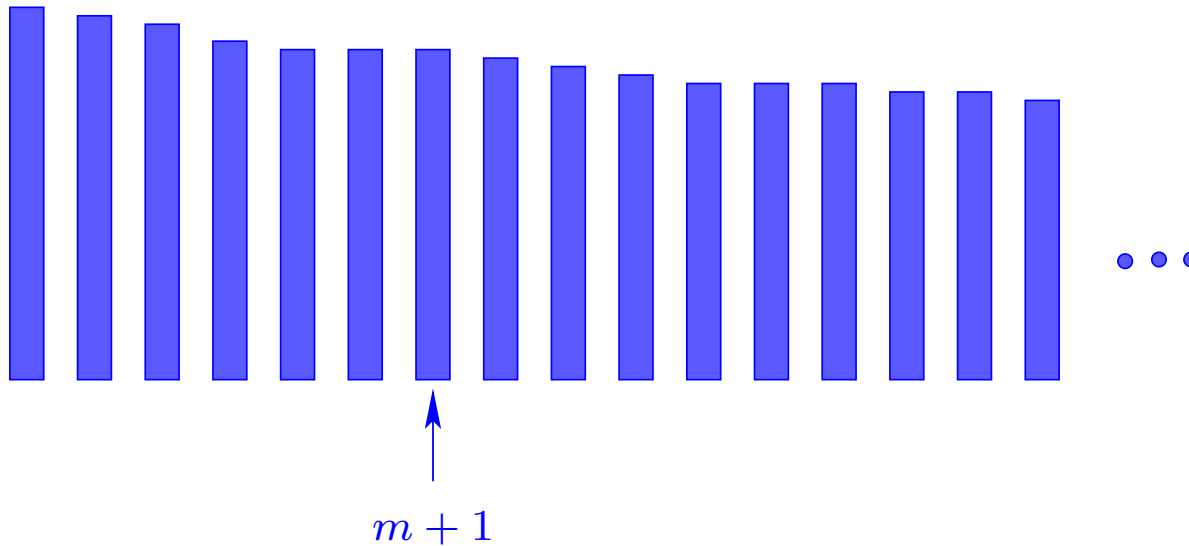
(a) $\frac{1}{m} \sum_{i=1}^n p_i$

(b) $\max_{1 \leq i \leq n} p_i$

(c) $2 \times$ proc. time of $(m + 1)$ -st largest job



Room for improvement?

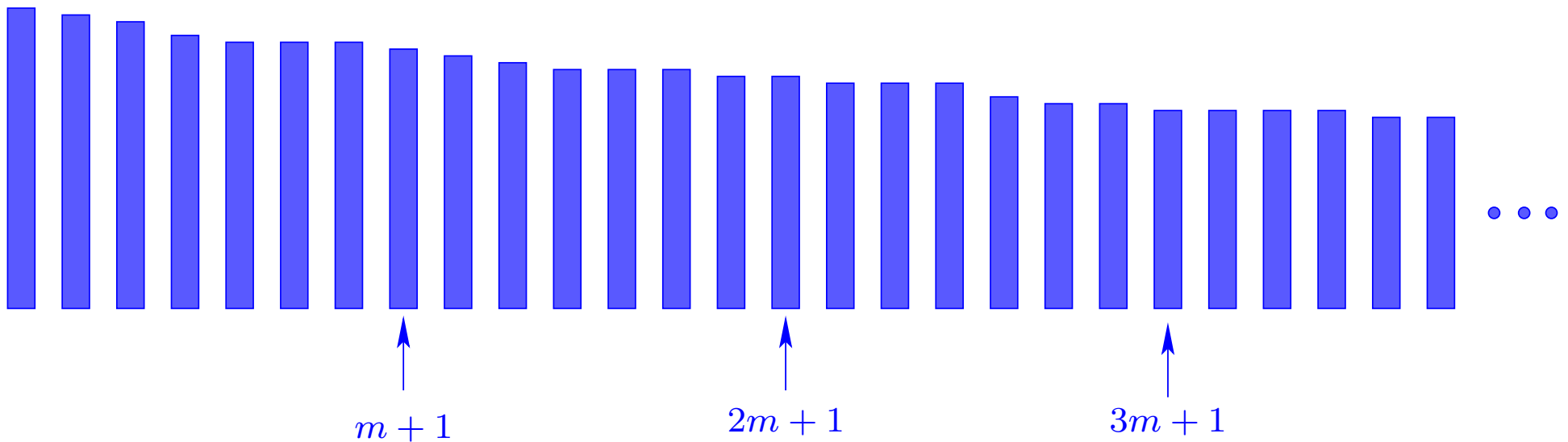


Thm: Impossible to prove competitive ratio smaller than 1.919 using

(1) $\frac{1}{m} \sum_{i=1}^n p_i$

(2) Set of $m + 1$ largest proc. times

Room for improvement?



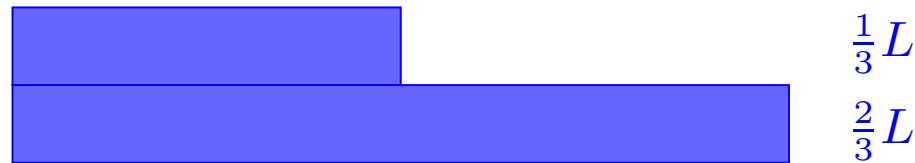
(3) $(im + 1)$ -st to $(im - i + 1)$ -st largest jobs, $i = 1 \dots, \lfloor \frac{(n-1)}{m} \rfloor$

Thm: Impossible to prove competitive ratio smaller than 1.917 using (1–3).

Randomization

- $m = 2$: 4/3-competitive algorithm. This is optimal.
Bartal, Fiat, Karloff, Vohra STOC 92
- Lower bound of $e/(e - 1) \approx 1.58$, for any m .
Chen, van Vliet, Woeginger IPL 94; Sgall IPL 97
- BIT algorithm that is 1.916-competitive, for any m .
Albers STOC 02

2 machines



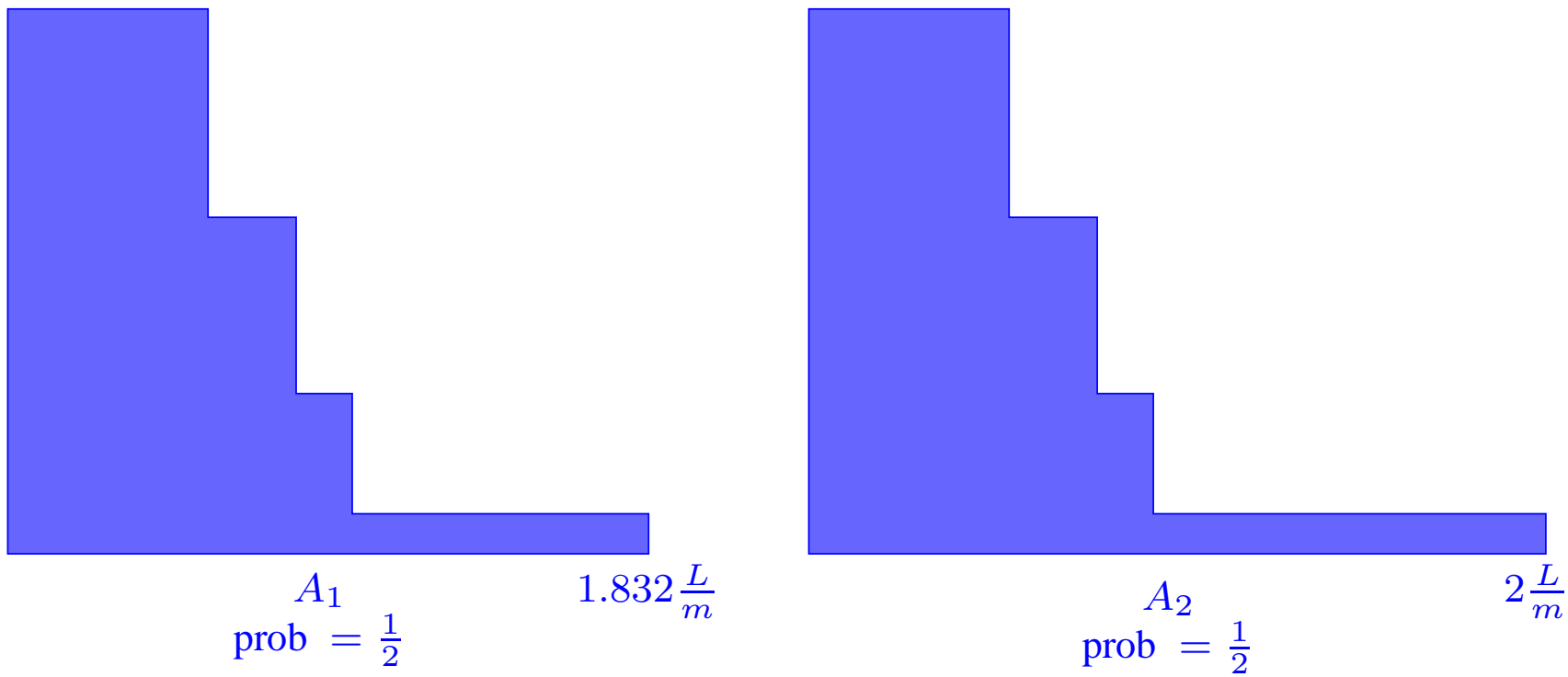
Discrepancy = load difference on the two machines

L = total proc. times of all jobs arrived so far.

Algorithm: Maintain set of all schedules generated so far, together with their probabilities. When a new job arrives:

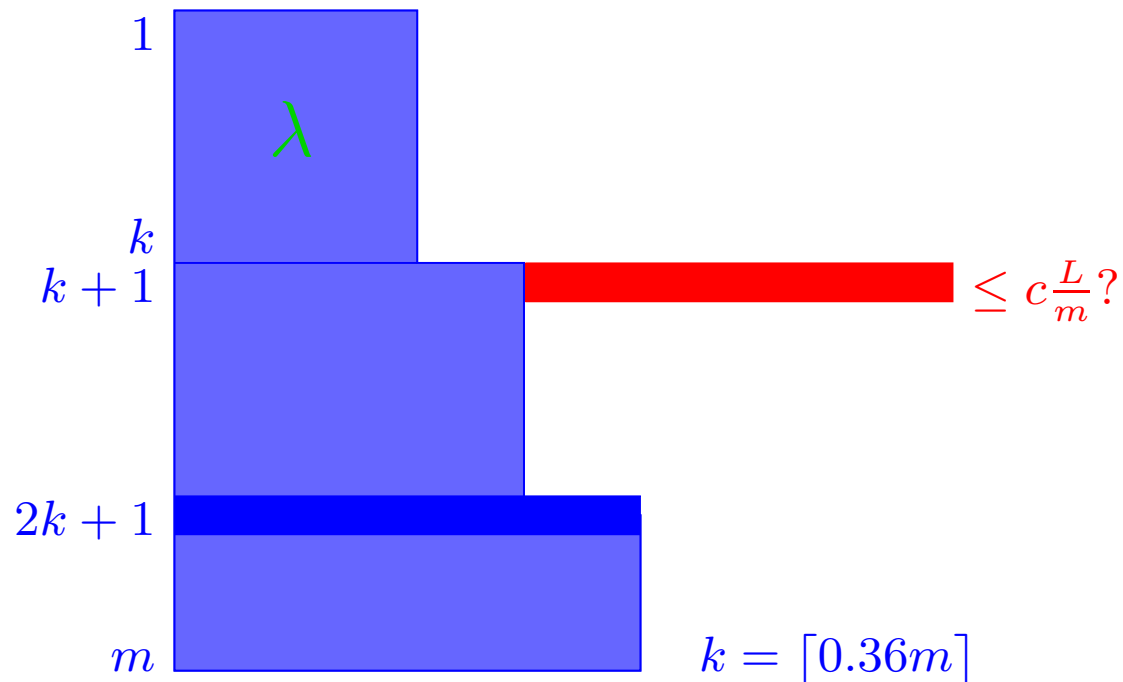
- E_1 = discrepancy if job is put on short machine in each schedule.
- E_2 = discrepancy if job is put on tall machine in each schedule.
- Determine p , $0 \leq p \leq 1$, such that $pE_1 + (1 - p)E_2 = \frac{1}{3}L$.
- If p exists, with prob. p schedule on short machine and with prob. $1 - p$ on the tall machine in each schedule.
- If such p does not exist, schedule on short machine.

BIT algorithm



$$\mathbf{E}[BIT(\sigma)] \leq \frac{1}{2}(1.832 + 2)\frac{L}{m} = 1.916\frac{L}{m}$$

Algorithm A_1



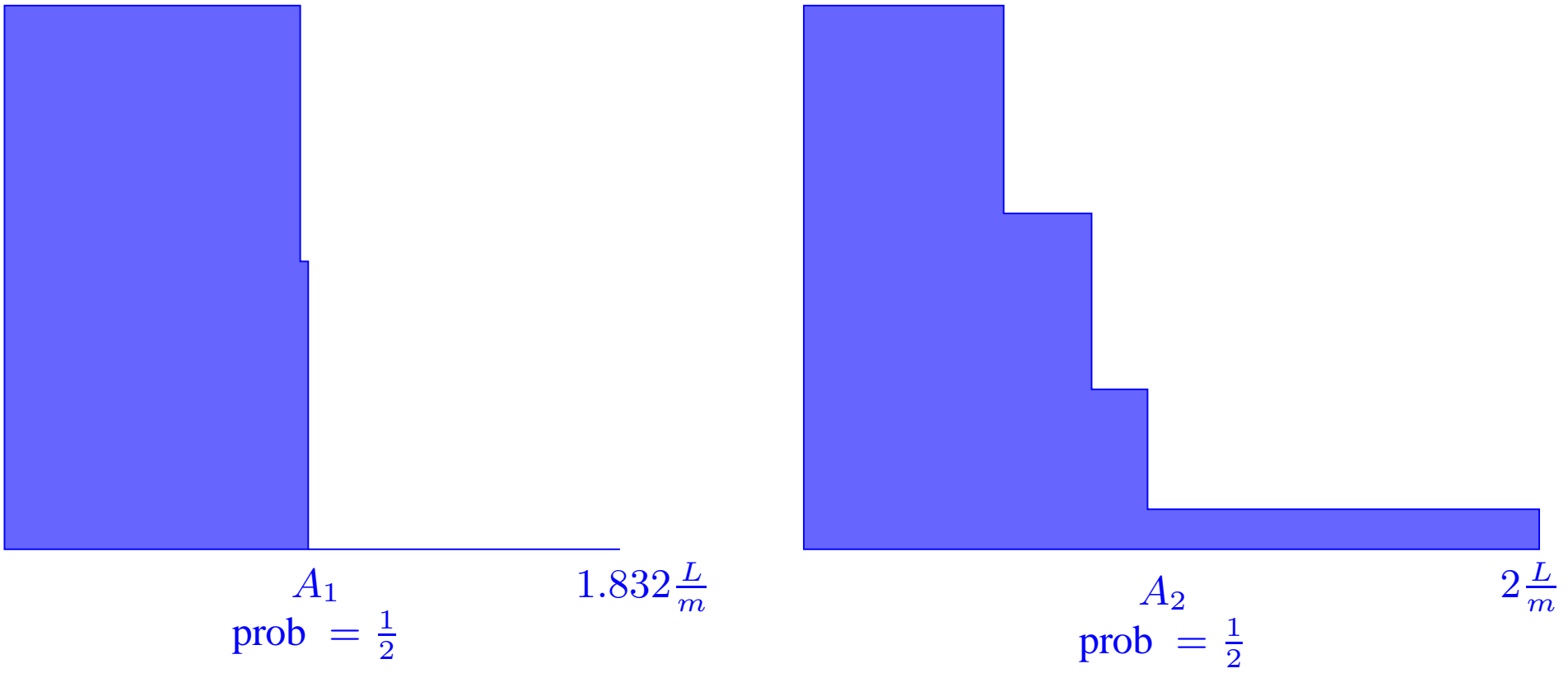
Algorithm: $c = 1.832$ $k = \lceil 0.36m \rceil$ $\alpha = 0.56$

Schedule new job on machine $k+1$ if

- $\lambda > \alpha \cdot l_{2k+1}$ and resulting load $\leq c \cdot \frac{L}{m}$

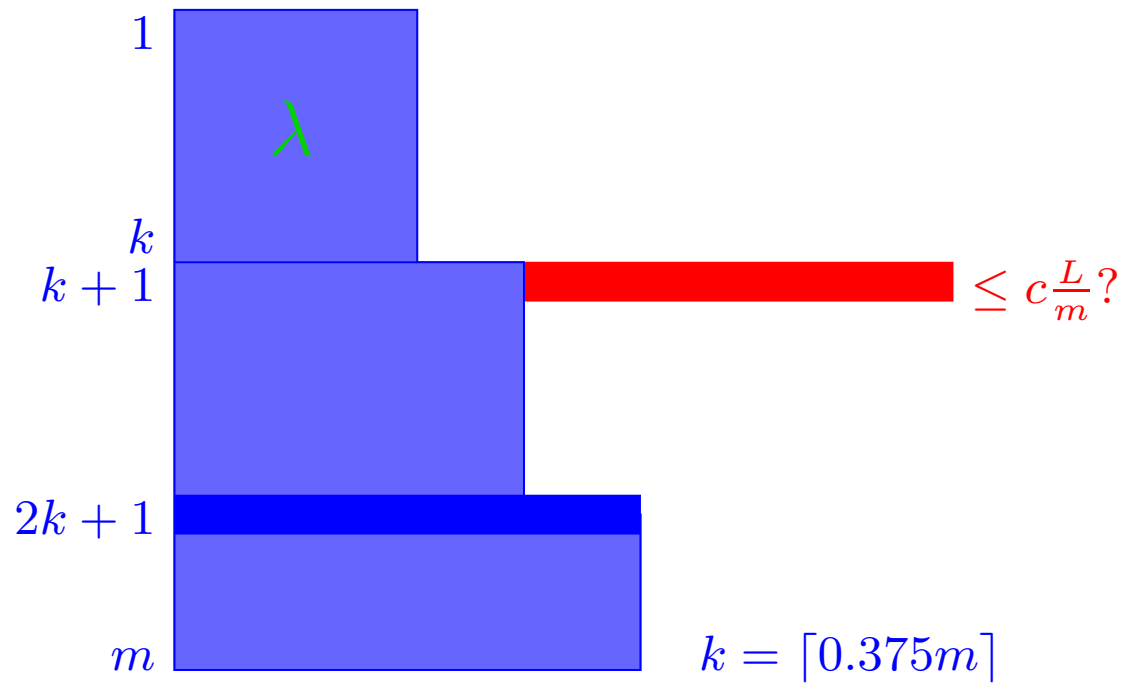
Otherwise schedule job on machine 1.

Problem



A_1 's schedule is nearly balanced/flat.

Algorithm A_2



Algorithm: $k = \lfloor 0.375m \rfloor$ $\alpha = 0.45$

Keep track of A_1 's schedule. If A_1 not flat then $c = 2$ else $c = 1.885$.

Schedule new job on machine $k+1$ if

- $\lambda > \alpha \cdot l_{2k+1}$ and resulting load $\leq c \cdot \frac{L}{m}$

Otherwise schedule job on machine 1.

Extensions

Preemption allowed: Tight upper and lower bounds of $e/(e - 1)$.

l_p norm: Minimize $(\sum_{i=1}^m l_i^p)^{1/p}$, where l_i is the load of i -th machine.

Motivation: Balance load among the machines.

Line breaks in LaTeX are computed based on l_2 and l_3 norms.

$p = \infty$: Makespan

$p = 2$: List is $\sqrt{4/3}$ -competitive.

Open problems

Develop improved **deterministic** algorithms.

Develop **randomized algorithms** that beat the deterministic lower bound, for all m .

Jobs arrive over time

Job sequence J_1, J_2, \dots, J_n

J_i : arrival time a_i

processing time p_i

possible weight w_i

preemption may be allowed

- **Clearvoyant scheduling:** When J_i arrives, p_i is known.
Classical manufacturing; Web server delivering static Web pages.
- **Non-clearvoyant scheduling:** When J_i arrives, p_i is unknown and becomes known only when job finishes.
Scheduling in operating systems.

Algorithms

Clearvoyant

- **SRPT:** Shortest Remaining Processing Time Run job with least remaining work.
- **SJF:** Shortest Job First Run job with least initial work.
- **HDF:** Highest Density First Run job with highest ratio w_i/p_i

Algorithms

Non-clearvoyant

- **FIFO**: First In First Out or First Come First Served. Run job with earliest arrival time.
- **RR**: Round Robin or Processor Sharing or Equi-Partition
Devote an equal amount of processing resources to all jobs.
Multi-processor environment: Assign J_i to processor $i \bmod m$.
- **SETF**: Shortest Elapsed Time First Run the job that has been processed the least. Amounts to RR when there are ties.
- **MLF**: Multi-Level Feedback Mimick SETF while keeping the number of preemptions per job logarithmic.
Queues Q_0, Q_1, \dots with target processing times $T_j = 2^j$.
 J_i is processed for $T_j - T_{j-1}$ time units while in Q_j and before being promoted to Q_{j+1} . Always run job in the front of lowest non-empty queue.

Clearvoyant scheduling

Job sequence J_1, J_2, \dots, J_n

J_i : arrival time a_i

processing time p_i

preemption allowed

Total flow time: $\sum_{i=1}^n f_i$

- 1 machine: SRPT is 1-competitive.
- m machines: SRPT is $O(\min\{\log P, \log n/m\})$ -competitive, where $P = p_{\max}/p_{\min}$. This ratio is best possible.

Resource augmentation: SPRT is $1/s$ -competitive using speed $s \geq 2 - \frac{1}{m}$.

No migration: $O(\min\{\log P, \log n/m\})$ -competitive algorithm.

Clearvoyant scheduling

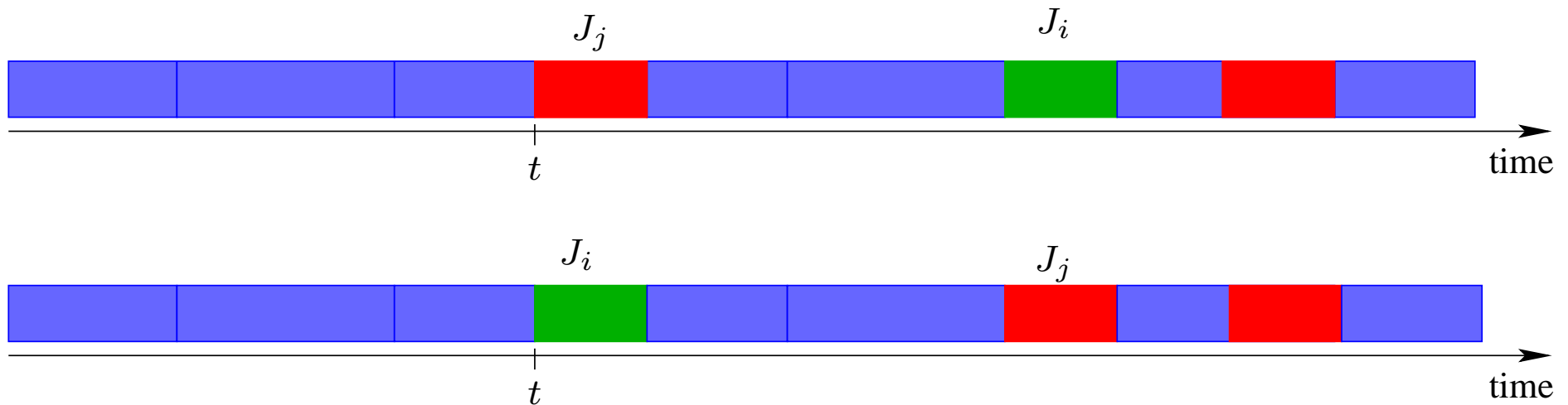
Total stretch: $\sum_{i=1}^n f_i/p_i$

- 1 machine: SRPT is 2-competitive.
Any algorithm has competitiveness $c \geq 1.036$.
 - m machines: SRPT is 14-competitive.
Any algorithm has competitiveness $1.093 \leq c \leq 9.82$.
- No migration: 17.32-competitive algorithm.

1 machine

SRPT is optimal.

Consider optimal schedule. Suppose that it violates SRPT first at time t .



Use slots where J_i or J_j are scheduled to first process J_i and then J_j .

Swap can only decrease the total flow time.

m machines

Thm: SRPT is $O(\log P)$ -competitive where $P = p_{\max}/p_{\min}$

For any scheduler S let F_S be the total flow time on given job instance.

$N_S(t)$ = #active jobs in S at time t

$M_S(t)$ = #active machines in S at time t

$$F_S = \int_t N_S(t) dt$$

Observation 1: $F_S \geq \sum_i p_i$

Observation 2: $\int_t M_S(t) dt = \sum_i p_i$

Analysis SRPT

T = times when all m machines are active.

Main Lemma: $t \in T \quad N_{SRPT}(t) \leq m(2 + \ln P) + N_{OPT}(t)$

$$F_{SRPT} = \int_t N_{SRPT}(t) dt = \int_{t \notin T} N_{SRPT}(t) dt + \int_{t \in T} N_{SRPT}(t) dt$$

Analysis SRPT

T = times when all m machines are active.

Main Lemma: $t \in T \quad N_{SRPT}(t) \leq m(2 + \ln P) + N_{OPT}(t)$

$$\begin{aligned} F_{SRPT} &= \int_t N_{SRPT}(t) dt = \int_{t \notin T} N_{SRPT}(t) dt + \int_{t \in T} N_{SRPT}(t) dt \\ &\leq \int_{t \notin T} M_{SRPT}(t) dt + \int_{t \in T} ((2 + \ln P) M_{SRPT}(t) + N_{OPT}(t)) dt \end{aligned}$$

Analysis SRPT

T = times when all m machines are active.

Main Lemma: $t \in T \quad N_{SRPT}(t) \leq m(2 + \ln P) + N_{OPT}(t)$

$$\begin{aligned} F_{SRPT} &= \int_t N_{SRPT}(t) dt = \int_{t \notin T} N_{SRPT}(t) dt + \int_{t \in T} N_{SRPT}(t) dt \\ &\leq \int_{t \notin T} M_{SRPT}(t) dt + \int_{t \in T} ((2 + \ln P) M_{SRPT}(t) + N_{OPT}(t)) dt \\ &\leq (2 + \ln P) \int_t M_{SRPT}(t) dt + F_{OPT} \leq (2 + \ln P) \sum_i p_i + F_{OPT} \end{aligned}$$

Analysis SRPT

Lemma: For any t and i with $N_{OPT}(t) + 2m + i \leq N_{SRPT}(t)$, the $(N_{OPT}(t) + 2m + i)$ -th largest remaining processing time of SRPT is at least $p_{\min} \left(\frac{m}{m-1}\right)^i$.

Main Lemma: $t \in T \quad N_{SRPT}(t) \leq m(2 + \ln P) + N_{OPT}(t)$

Proof Main Lemma: $p_{\min} \left(\frac{m}{m-1}\right)^i \leq p_{\max}$ and hence

$$i \leq \log_{m/(m-1)} P$$

$$N_{SRPT}(t) \leq N_{OPT}(t) + 2m + \log_{m/(m-1)} P \leq N_{OPT}(t) + 2m + m \ln P$$

since $\log_{m/(m-1)} P = \log_{(m/(m-1))^m \frac{1}{m}} P$

Analysis SRPT

Lemma: For any t and i with $N_{OPT}(t) + 2m + i \leq N_{SRPT}(t)$, the $(N_{OPT}(t) + 2m + i)$ -th largest remaining processing time is at least $p_{\min} \left(\frac{m}{m-1}\right)^i$.

Proof:

- $VOL_A(t, x) =$ total volume of remaining processing time $\leq x$.
 $VOL_{SPRT}(t, x) - VOL_{SPRT}(t, x) \leq mx$
- Prove lemma for $N_{OPT}(t) = 0$.
- Any additional job in $N_{OPT}(t)$ can increase $N_{SRPT}(t)$ by only 1.

Eliminating migration

$O(\min\{\log P, \log n\})$ -competitive algorithm

- Job classes: class k contains jobs with initial processing time in $[2^k, 2^{k+1})$.
- When new job of class k arrives, assign it to machine that was assigned the smallest load of class- k jobs (List scheduling).
- On each processor execute SRPT.

Non-clearvoyant scheduling

Job sequence J_1, J_2, \dots, J_n

J_i : arrival time a_i

processing time p_i **unknown**

preemption allowed

1 machine, total flow time: $\sum_{i=1}^n f_i$

- Any deterministic algorithm is $\Omega(n^{1/3})$ -competitive.
- **SETF is $(1 + 1/\epsilon)$ -competitive** using speed $1 + \epsilon$.
- Randomized algorithms: $\Theta(\log n)$ -competitive.

Load balancing

m machines, **restricted machine model**

J_1, \dots, J_n

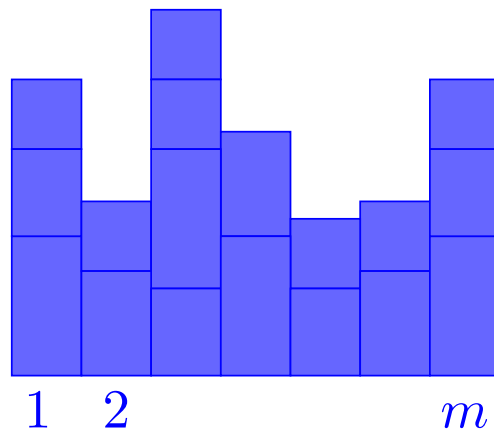
J_i : arrival time a_i

load l_i

unknown duration

executable on any machine in $M_i \subseteq \{1, \dots, m\}$

Goal: min. maximum load on the machines



Load balancing

Thm: Any det. algorithm is $\Omega(\sqrt{m})$ -competitive even if all loads are 1.

Azar, Broder, Karlin FOCs 92

Thm: Robin-Hood is $(2\sqrt{m} + 2)$ -competitive.

Azar, Kalyanasundaram, Plotkin, Pruhs, Waarts J. Algorithms 97

Lower bound

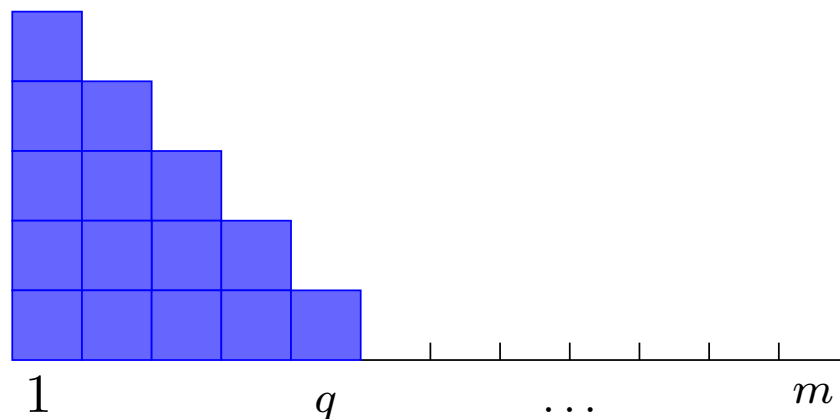
Thm: A online algorithm $\implies c_A \geq \Omega(\sqrt{m})$

Proof: $L_j(t)$: A 's load of j -th machine just prior to arrivals at t

Order machines s.t. $L_1(t) \geq \dots \geq L_{q(t)} \quad L_j(t) = 0 \quad j > q(t)$

Construct job sequence s.t. m active jobs

$$\text{OPT's load} = 1 \quad L_j(t) \geq L_{j+1}(t) + 1 \quad 1 \leq j \leq q - 1$$

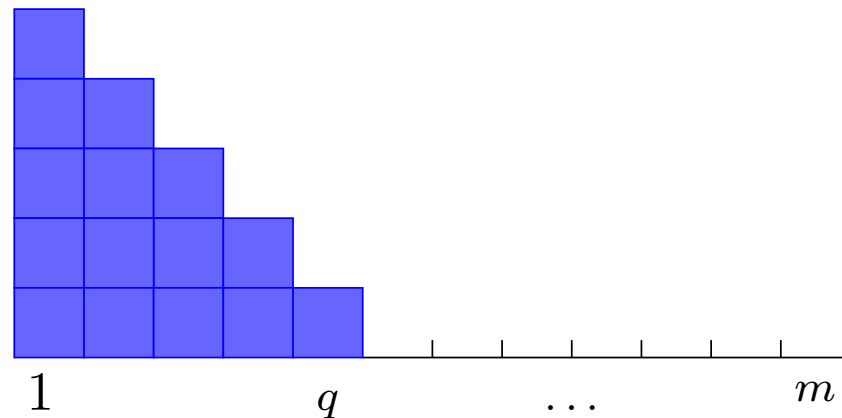


Lower bound

Construct job sequence s.t. m active jobs

OPT's load = 1

$$L_j(t) \geq L_{j+1}(t) + 1 \quad 1 \leq j \leq q - 1$$



$$m = \sum_{j=1}^q L_i(t) \leq \sum_{j=0}^{q-1} (l - j) \leq \frac{l(l+1)}{2} \leq l^2$$

because $q \leq l$ and $l \geq 1$.

$$l \geq \sqrt{m}$$

Construction

Consider

$$L = (L_1(t), \dots, L_{q(t)}(t))$$

$$L' = (L_1(t'), \dots, L_{q(t')}(t'))$$

$L < L'$ if $L_k(t) = L_k(t')$ for $k < j$ and $L_j(t) < L_j(t')$

Stages

Base case:



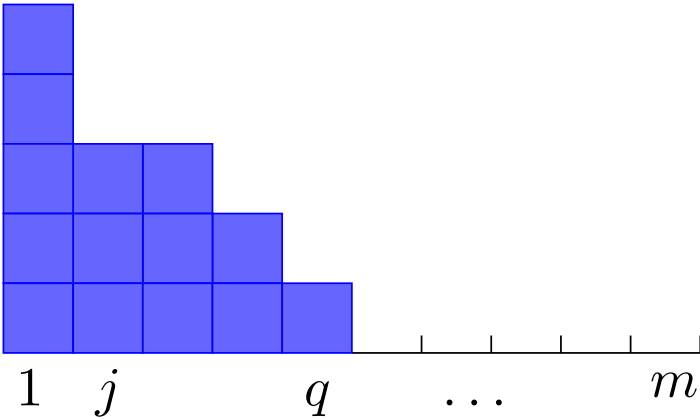
Construction

Inductive step:

At the end of k -th stage $L^k = (L_1^k, \dots, L_{q_k}^k)$

If $L_j^k \geq L_{j+1}^k + 1$, we are done. Otherwise construct L^{k+1} with $L^{k+1} > L^k$

Invariant: OPT has job i on machine j and $L_j^k > 0$, then A has i on j .



A



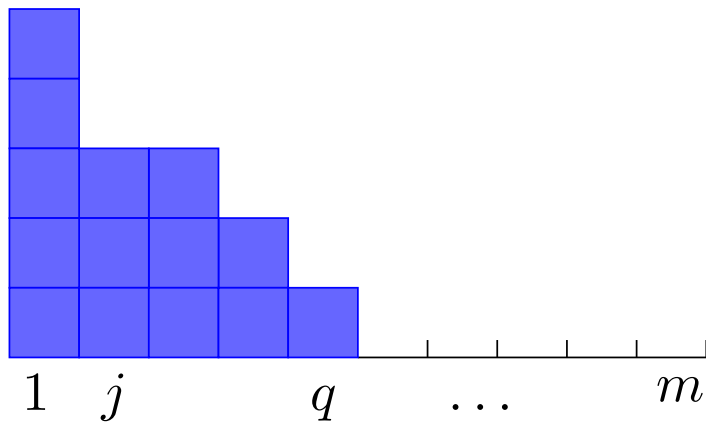
OPT

Determine smallest j with $L_j^k = L_{j+1}^k$

Construction

Inductive step:

Invariant: OPT has job i on machine j and $L_j^k > 0$, then A has i on j .



A



OPT

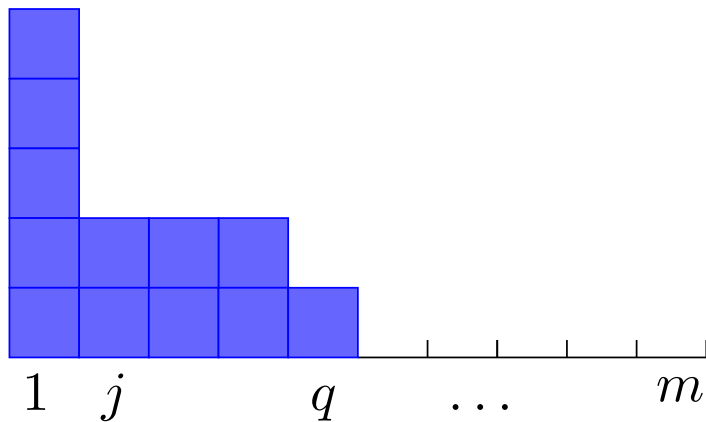
Determine smallest J with $L_j^k = L_{j+1}^k$

- Terminate OPT's jobs on $j, j + 1$
- Create J with allowable machines $\{j, j + 1\}$.
Assume A assigns to j . Then OPT assigns to $j + 1$.

Construction

Inductive step:

Invariant: OPT has job i on machine j and $L_j^k > 0$, then A has i on j .



A



OPT

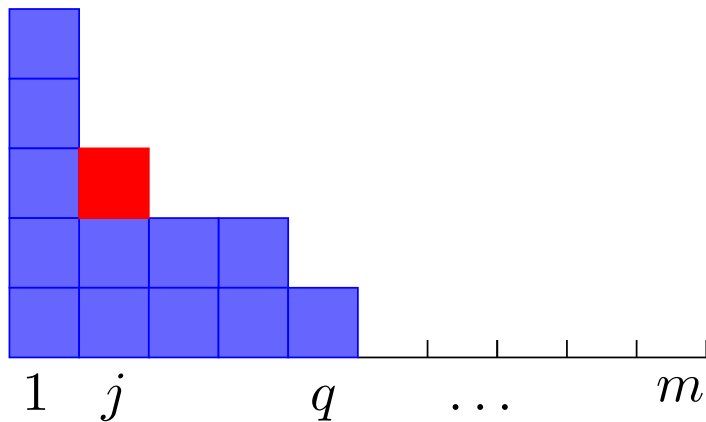
Determine smallest j with $L_j^k = L_{j+1}^k$

- **Terminate** OPT's jobs on $j, j + 1$
- Create J with allowable machines $\{j, j + 1\}$.
Assume A assigns to j . Then OPT assigns to $j + 1$.

Construction

Inductive step:

Invariant: OPT has job i on machine j and $L_j^k > 0$, then A has i on j .



A



OPT

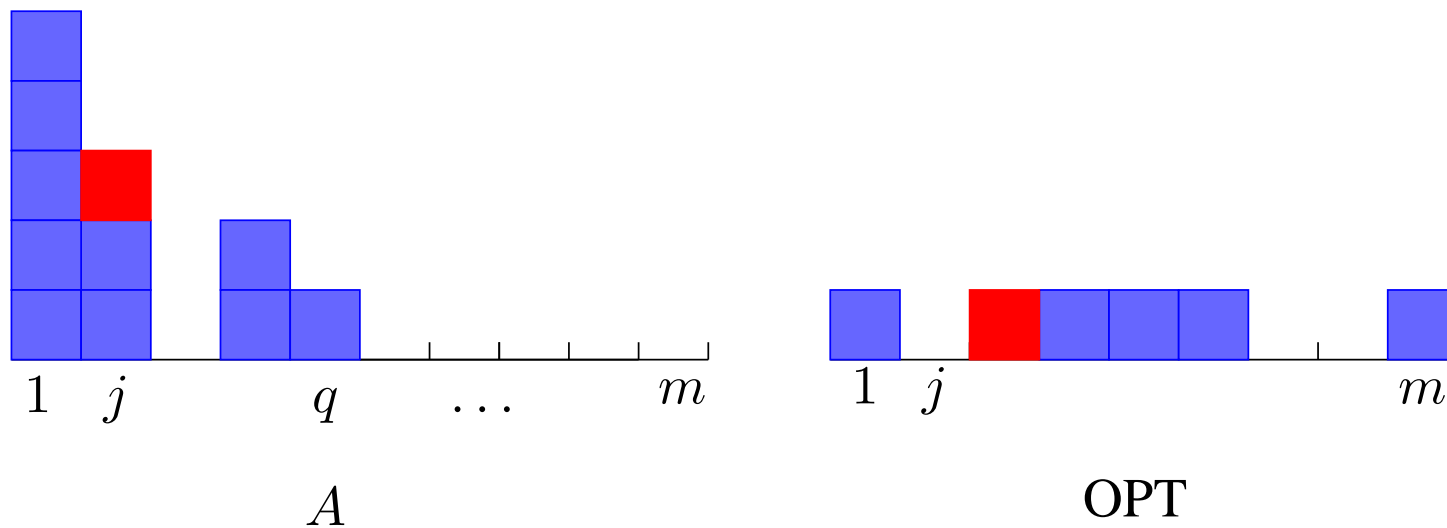
Determine smallest j with $L_j^k = L_{j+1}^k$

- Terminate OPT's jobs on $j, j + 1$
- Create J with allowable machines $\{j, j + 1\}$.
Assume A assigns to j . Then OPT assigns to $j + 1$.

Construction

Inductive step:

Invariant: OPT has job i on machine j and $L_j^k > 0$, then A has i on j .

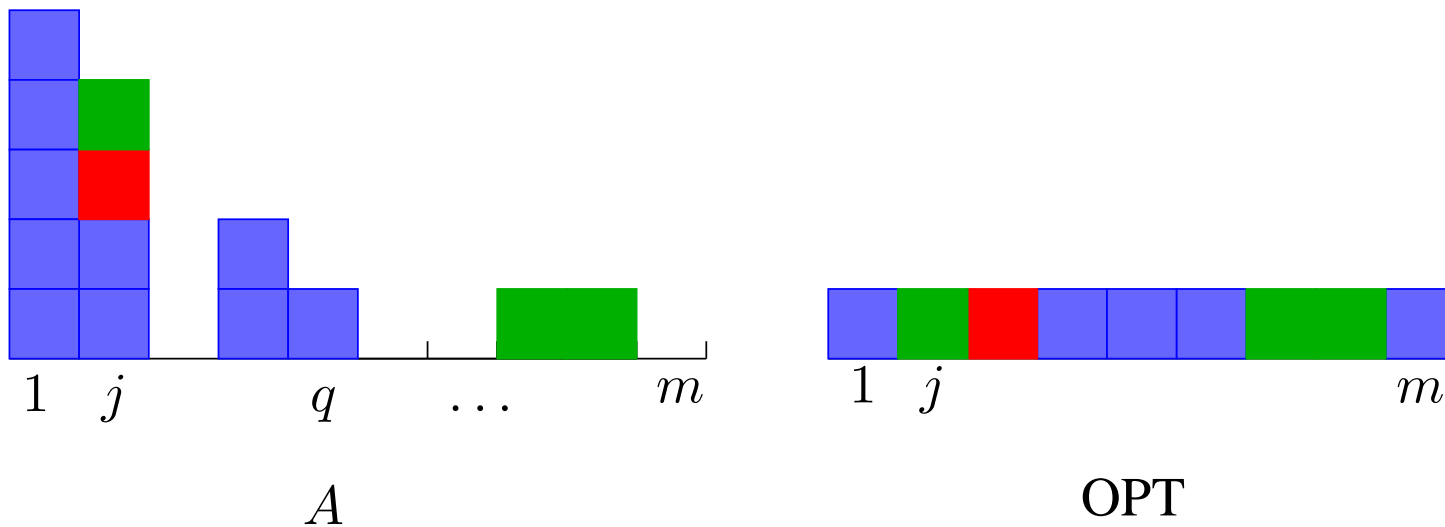


- Terminate A 's jobs on $j + 1$.
- For each of OPT 's idle machines, create job that can only be executed on that machine.

Construction

Inductive step:

Invariant: OPT has job i on machine j and $L_j^k > 0$, then A has i on j .



- Terminate A 's jobs on $j + 1$.
- For each of OPT 's idle machines, create job that can only be executed on that machine.

Algorithm Robin Hood

$$\sigma_i = J_1, \dots, J_i$$

$L_j(a_i)$: load on machine j just prior to a_i

Lower bound on $\text{OPT}(\sigma_i)$

$$B(a_0) = 0 \quad B(a_i) = \max\left\{B(a_{i-1}), l_i, \frac{1}{m}\left(l_i + \sum_j L_j(a_i)\right)\right\}$$

Machine is **rich** if $L_j(a_i) \geq \sqrt{m}B(a_i)$ and **poor** otherwise.

ROBIN HOOD: Assign J_1 to any machine.

J_i : Assign to **poor** machine if possible. Otherwise to rich machine that **most recently became rich**.

Thm: ROBIN HOOD is $(2\sqrt{m} + 2)$ -competitive.

Proof

At any time most \sqrt{m} rich machines.

$$mB(a_i) \geq \sum_j L_j(a_i)$$

If there were more than \sqrt{m} rich machines, then $\sum_j L_j(a_i) > \sqrt{m}\sqrt{m}B(a_i)$.

J_i arrives and assigned to machine j_0

1. j_0 poor

$$L_{j_0}(a_i) + l_i < \sqrt{m}B(a_i) + \text{OPT}(\sigma_i) \leq (\sqrt{m} + 1)\text{OPT}(\sigma_i)$$

Proof

2. j_0 rich

$a_{t(i)}$: last time when j_0 became rich

$S = \{\text{jobs assigned to } j_0 \text{ during } (a_{t(i)}, a_j)\}$

$k \in S$ could only be assigned to machines that were rich at time $a_{t(i)}$

$$h = \left| \bigcup_{k \in S} M_k \right| \leq \sqrt{m}$$

$$\sum_{k \in S} l_k \leq h \text{OPT}(\sigma_i) \leq \sqrt{m} \text{OPT}(\sigma_i)$$

$$\begin{aligned} L_{j_0}(a_i) + l_i &\leq L_{j_0}(a_{t(i)}) + l_{t(i)} + \sum_{k \in S} l_k + l_i < \sqrt{m}B(a_{t(i)}) + l_{t(i)} + \sum_{k \in S} l_k + l_i \\ &\leq (2\sqrt{m} + 2)\text{OPT}(\sigma_i) \end{aligned}$$

Summary

- **Online settings:** jobs form a list; jobs arrive over time
- **Classical and new objectives:** makespan, total flow, total stretch, l_p norms, load balancing
- **Old and new algorithms:** List, SRPT, more sophisticated strategies
- Power of randomization and resource augmentation