Online load balancing

- Problem definition — **Offline results**
- Identical machines — PTAS [HS87]
  - R(Greedy)=$2-1/m$
  - Improvements (best $= 1.92$)
- Related machines — PTAS [HS88]
  - R(Greedy)=$\Theta(\log m)$
  - 8-competitive algorithm
- Restricted machines — PTAS for fixed m [HS76]; 2-approximation; nothing better than 3/2
  - R(Greedy)=$\Theta(\log m)$
  - This is best possible

The Problem

- m identical machines
- n nonpreemptable jobs
- jobs arrive one by one
- goal: minimize the *makespan*, the time at which the last job finishes
- load balancing

List Scheduling (LS)

LS assigns each job to the least loaded machine

Also known as the Greedy algorithm

It achieves a competitive ratio of

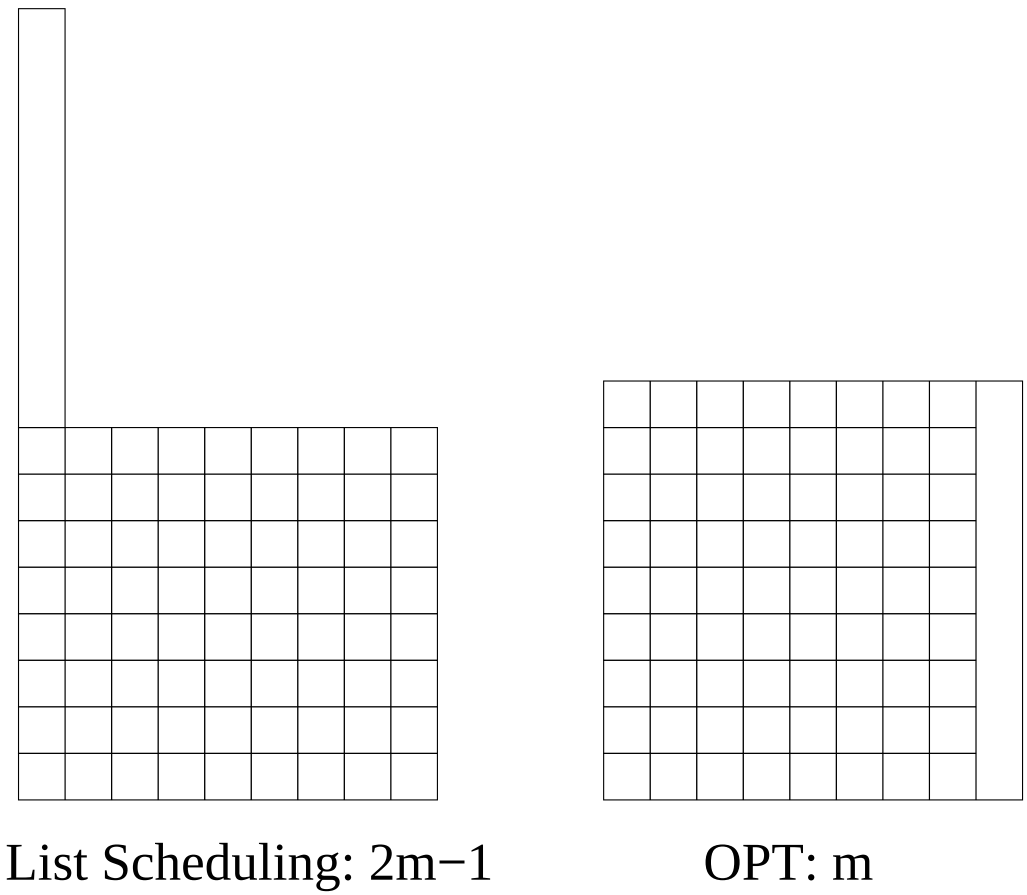
$$2 - \frac{1}{m}$$

This bound is tight for LS

Does there exist a better algorithm?

Lower bound for LS

Input: $m(m-1)$ jobs of size 1 and one job of size m.



This shows the competitive ratio is not better than $2-1/m$.

Lower bound for $m = 2$

Denote the optimal competitive ratio for m by $C(m)$. Let the job sequence $\sigma = \{1, 1, 2\}$.

First two jobs on the same machine $\Rightarrow C_A = 2$.

Otherwise $A(\sigma) = 3$ and $OPT(\sigma) = 2$. We have

$$C(2) \geq 3/2.$$

For $m = 2$, the competitive ratio of LS is $2 - \frac{1}{m} = 3/2$.

Thus

$$C(2) = 3/2.$$

The case $m = 3$

Let $\sigma = \{1, 1, 1, 3, 3, 3, 6\}$.

Case 1: First three jobs not on three different machines:
sequence ends, competitive ratio is 2 ($OPT(\sigma) = 1, A(\sigma) = 2$)

Case 2: Second three jobs not on three different machines:
sequence ends, competitive ratio is 7/4 ($OPT(\sigma) = 4, A(\sigma) = 7$)

Case 3: Else, after final job, load is 10 and $OPT(\sigma) = 6$.

This proves

$$C(3) \geq \frac{5}{3}.$$

Competitive ratio of LS is $5/3$ for $m = 3$, so LS is optimal

The case $m \geq 4$

Use as input sequence

$$\sigma = \{1, \overset{m}{\ldots}, 1, 1+\sqrt{2}, \overset{m}{\ldots}, 1+\sqrt{2}, 2+2\sqrt{2}\}$$

to prove

$$C(m) \geq 1 + \frac{1}{2}\sqrt{2} \qquad m = 4, 5, \ldots$$

Again, all jobs of same size must be placed on different machines.

Why does this not work for smaller m?

More on the case $m \geq 4$

- Modified List Scheduling
- A better lower bound
- Open questions

Modified List Scheduling: definitions

Let $1 \leq \beta \leq 3/2$ and define the symmetric relation

$$x \sim y \quad \Longleftrightarrow \quad \frac{y}{\beta} \leq x \leq \beta y$$

and say that in this case x is similar to y.

$\sim(S)$, where S is a set, means all elements in S are similar.

Idea of MLS: maintain some imbalance, try to prevent the machines from becoming similar.

Let

$$R = \frac{2m-2+\beta}{m-1+\beta}.$$

Modified List Scheduling (MLS)

Read_job (x); while x $\neq$ End do{

if $\not\sim (L_1 + x, L_2, \ldots, L_m)$ then

Assign (x, 1)

else

if $L_2 + x \leq R \frac{\sum_i L_i + x}{m}$ then

Assign (x, 2)

else

Assign (x, 1);

Order the machines such that $L_1 \leq L_2 \leq \cdots \leq L_m$;

Read_job (x);

};

Competitive ratio of MLS

- MLS improves upon LS for all $m \geq 4$
- $\lim_{m\to\infty} C_{MLS} = 2$

For $m = 4$ we find $C_{MLS} = 1.7333$ (and $C_{LS} = 1.75$)

Proofs are omitted.

Later algorithms are better than 2-competitive also in the limit (best known result is 1.92)

A better lower bound for $m = 4$

Idea is similar to the lower bound for $m = 2, 3$, but the job sizes are not straightforward.

Job sequence uses parameters x and y

$$\begin{aligned}\sigma = \{ & 1, 1, 1, 1, \\ & x, x, x, x, \\ & y, y, y, y, \\ & 3x+2y+2, 3x+2y+2, 3x+2y+2, 5x+2y+2, \\ & 6x+5y+4\}\end{aligned}$$

Choosing x and y, we can get a lower bound of 1.731.

Better lower bounds

Similar sequences can be used to show good lower bounds for $m = 5, 6, \ldots, 10$.

For $m = 4$, a MUCH longer sequence shows a lower bound of

$$\sqrt{3} \approx 1.732$$

(SIAM Journal on Computing, 2003)

Note that the current upper bound for $m = 4$ is 1.733.

Open questions

- What is the exact value of $C(4)$? We know that

$$\sqrt(3) \leq C(4) \leq 1.7333.$$

  The lower bound is 5 years old, the upper bound 10 years

Conjecture $C(4) = \sqrt{3} = 1.7320508\ldots.$

- Is $C(m) \leq C(m+1)$ for all m?
- What is the value of $\lim_{m\to\infty} C(m)$? At most 1.920.

Related machines

- So far we only considered *identical* machines
- All machines have the same speed
- We now turn to related machines
- Each machine has a speed
- The greedy algorithm is $\Theta(\log m)$-competitive
- We present a constant-competitive algorithm

Lower bound for the greedy algorithm

- We show a lower bound for 11 machines
- It can be extended to larger numbers
- The set of machines is as follows

4   2   2   1   1   1   1   1   1   1   1

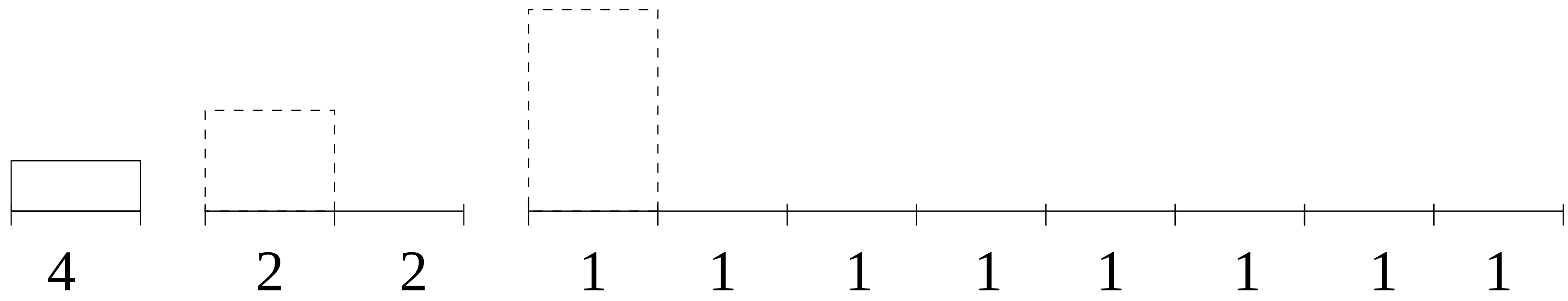
The set of jobs has sizes which match these speeds

Thus, OPT = 1

The smallest jobs (size 1) arrive first

Lower bound for the greedy algorithm

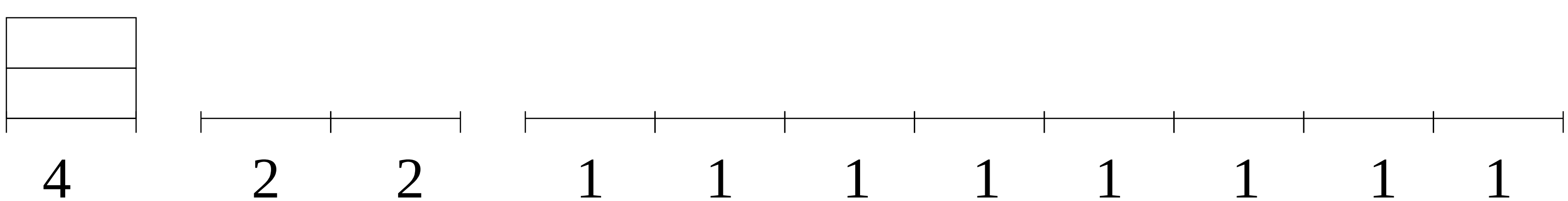
The first job goes on the fastest machine



Lower bound for the greedy algorithm

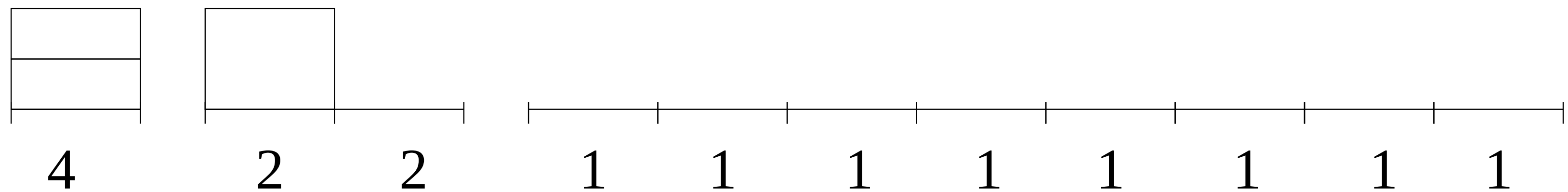
We may assume the speed “2” is actually $2 - \varepsilon$

Then the second job also goes on the fastest machine



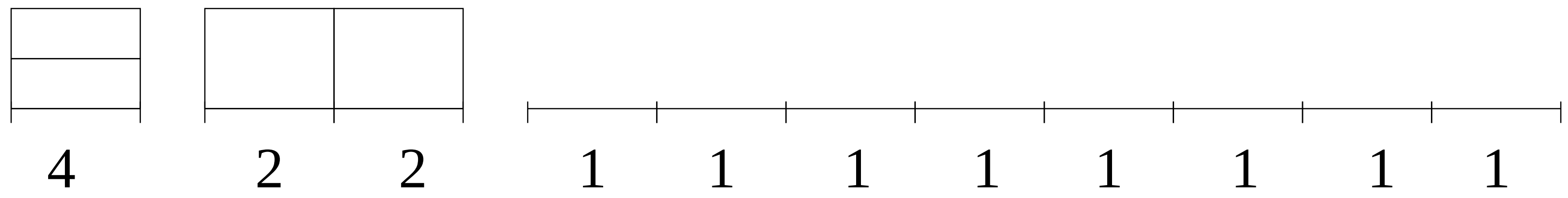
Lower bound for the greedy algorithm

The next job goes on a machine of speed 2



Lower bound for the greedy algorithm

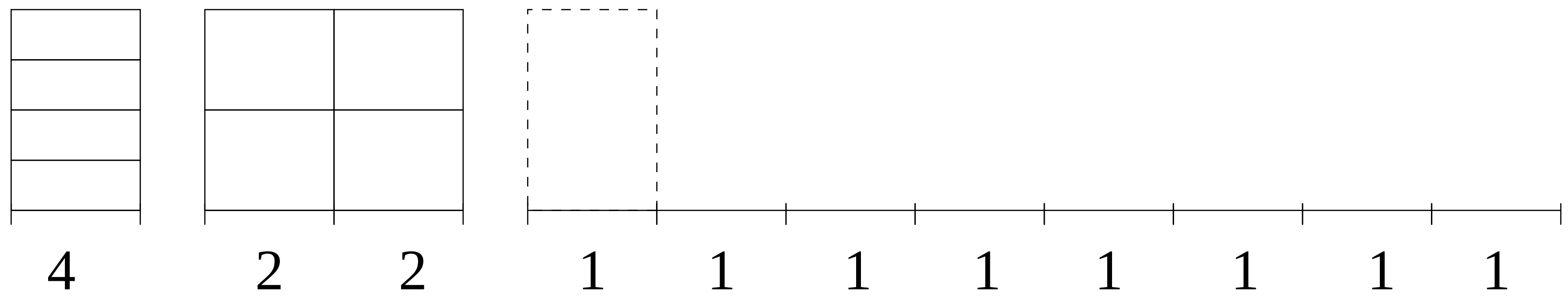
…and the next job as well



Lower bound for the greedy algorithm

The next four jobs are placed similarly

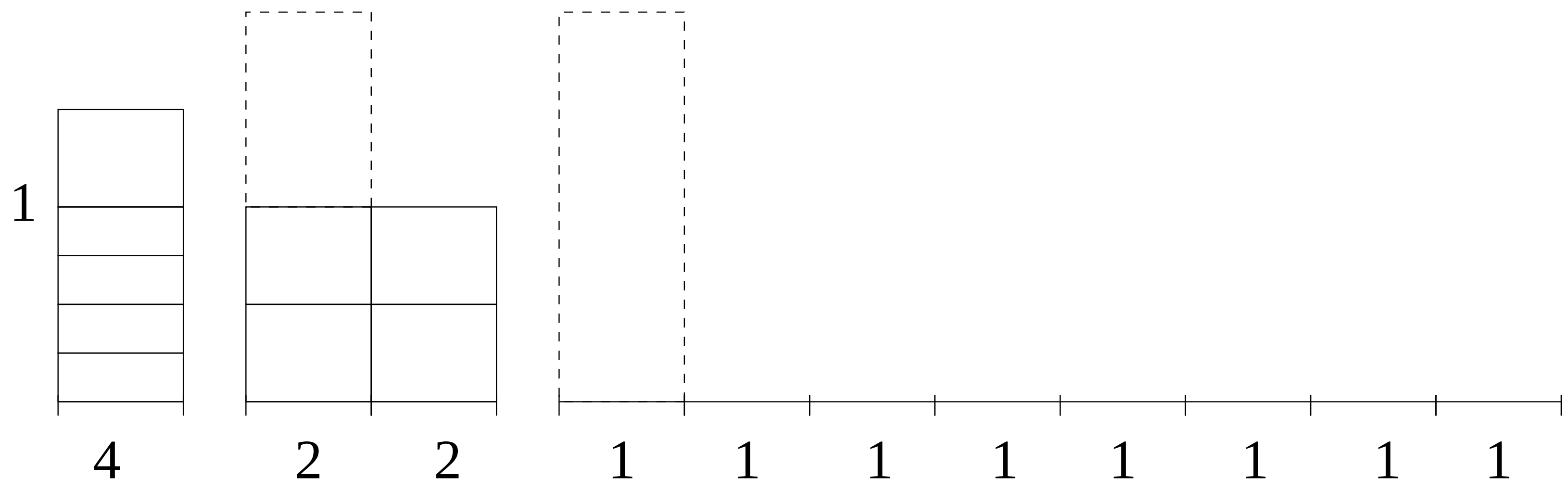
The slow machines do not get used for these jobs



Lower bound for the greedy algorithm

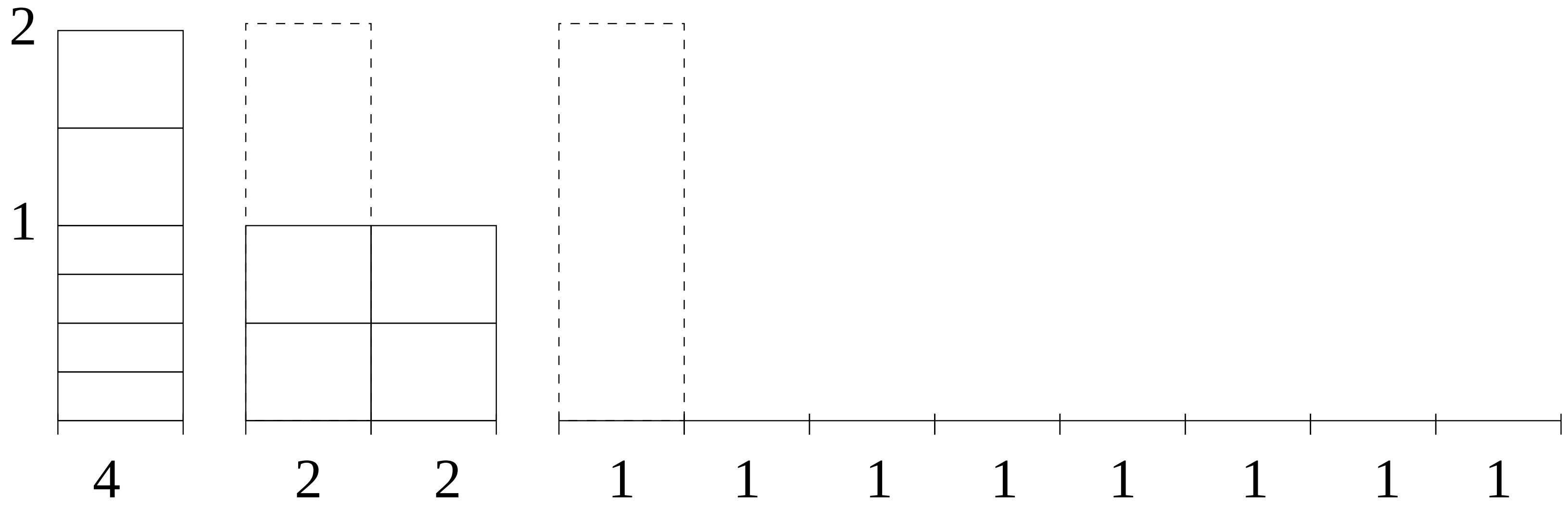
Now jobs of size 2 start to arrive

All jobs are placed on the machine where they complete the earliest



Lower bound for the greedy algorithm

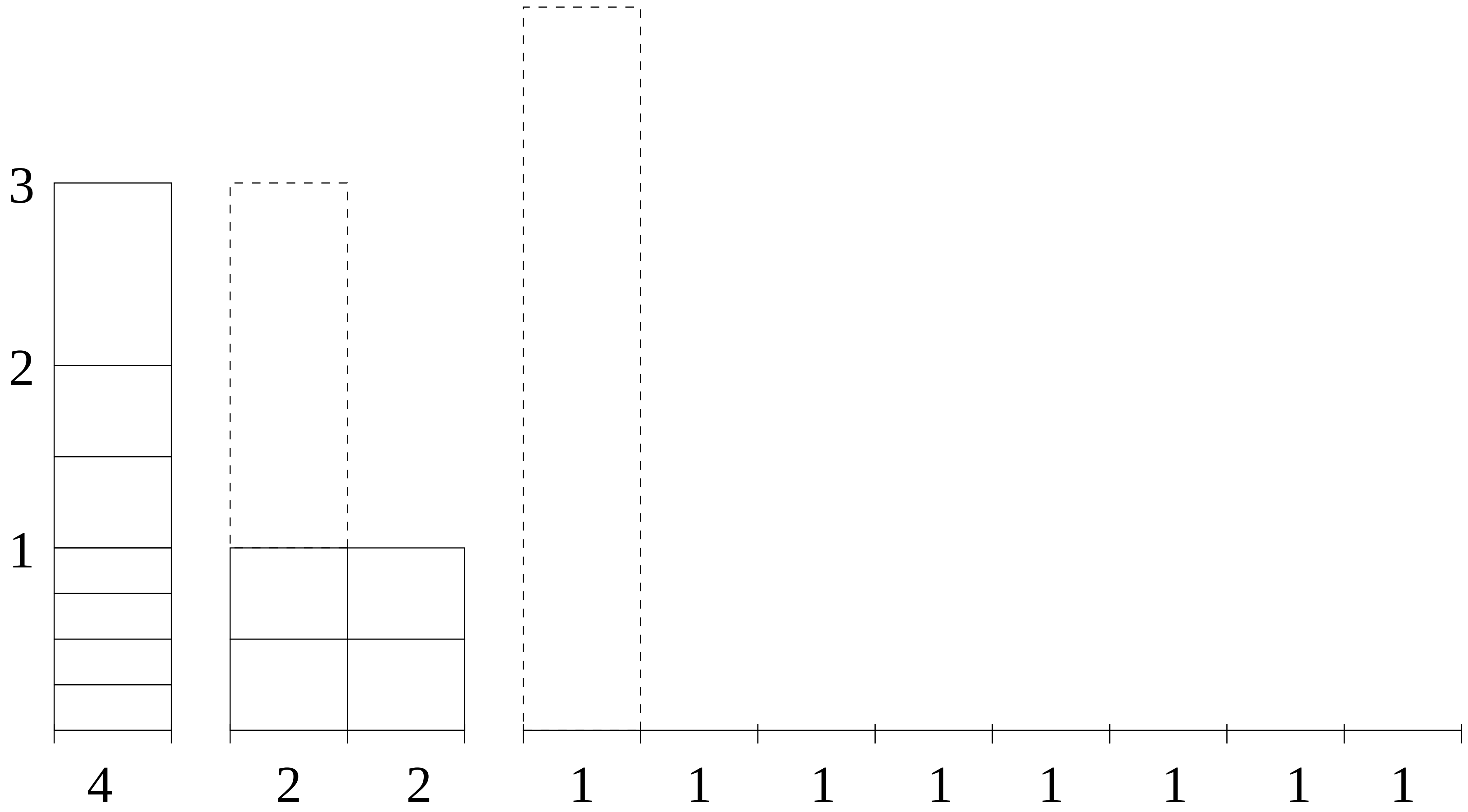
Both jobs of size 2 are placed on the fastest machine



Lower bound for the greedy algorithm

Final load is 3 = number of classes of machines

For general m, final load is $\Omega(\log m)$



The algorithm SLOWFIT

- We first present an algorithm that knows the optimal load
- We then show how to extend this to the general case
- Essentially, we simply guess OPT
- We double our guess when it it clear that it is too small
- This gives a constant competitive algorithm

An algorithm that knows OPT

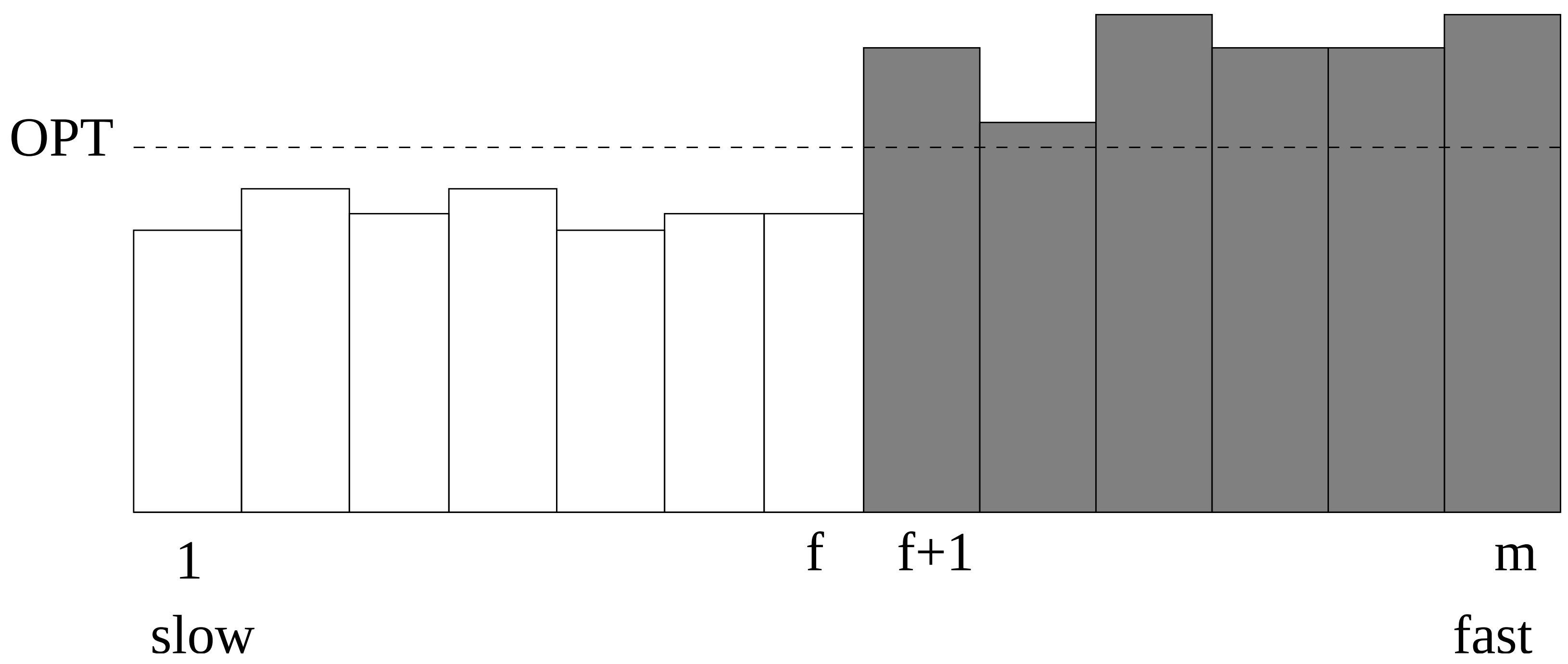
- Suppose OPT $\leq \Lambda$
- Order the machines by speed (M_1 is the slowest)
- Let new job request be r
- Put job on slowest machine where load remains below 2Λ
- If there is no such machine, output "failure"

This algorithm does not fail

- Suppose it fails on some input $\sigma = \{r_1, \ldots, r_n\}$
- Job r_n cannot be assigned
- Let f be the fastest machine with load below $\mathrm{OPT}(\sigma)$
- If $f = m$, r_n can be assigned to machine m
- Thus $f < m$

This algorithm does not fail

- The machines $f+1,\ldots,m$ are "overloaded"
- Let S_i be set of jobs assigned to machine i by this algorithm
- Let S_i^* be set of jobs assigned to machine i by OPT



An overloaded machine i

$$\begin{aligned}
\sum_{j\in S_i} p_j &= s_i \sum_{j\in S_i} \frac{p_j}{s_i} && \text{algebra} \\
&> s_i \cdot \mathrm{OPT}(\sigma) && \text{assumption} \\
&\geq s_i \sum_{j\in S_i^*} \frac{p_j}{s_i} && \text{definition } S_i^* \\
&= \sum_{j\in S_i^*} p_j && \text{algebra}
\end{aligned}$$

We have strict inequality for every overloaded machine

Thus, not all machines can be overloaded

An overloaded machine i (2)

- We have $\sum_{j \in S_i} p_j > \sum_{j \in S_i^*} p_j$
- There must be some job x on some overloaded machine i that OPT assigns to a slower machine $i' \leq f$
- The load of x on f would be less than $\mathrm{OPT}(\sigma)$ (OPT has x on a machine which is not faster than f)
- The load of machine f is still less than $\mathrm{OPT}(\sigma)$ at the end
- Our algorithm would assign x to f!
- Contradiction

The algorithm SLOWFIT (2)

- At start, set $\Lambda_0 = p_1/s_m$ (cost of OPT)
- Run previous algorithm until it fails
- Then, double Λ and continue
- In phase j, $\Lambda_j = 2^j \Lambda_0$
- In each phase, all previous assignments are ignored (machines are assumed to be empty)

If there is only one phase, this algorithm is 2-competitive

Analysis of SLOWFIT

- Suppose SLOWFIT terminates in phase $h > 0$
- Then the subroutine failed in phases $1, \ldots, h-1$
- Let σ_j be the request sequence in phase j
- This implies $\text{OPT}(\sigma_{h-1} r) > \Lambda_{h-1}$ (r is first request of phase h)
- Therefore $\text{OPT}(\sigma) > 2^{h-1}\Lambda_0$
- The makespan of SLOWFIT is at most

$$\sum_{j=0}^{h} \text{ALG}(\sigma_j) \leq \sum_{j=0}^{h} 2 \cdot 2^j \Lambda_0 = 2 \cdot (2^{h+1} - 1)\Lambda_0 < 8 \cdot \text{OPT}(\sigma)$$

Restricted machines

- This is a special case of unrelated machines
- Machines do not have speeds
- Instead, the load of a job depends on the machine that it is assigned to
- Each job is represented as a vector of loads
- For restricted machines, the load of job k is either w_k or infinite

The greedy algorithm on restricted machines

- For job k, we call the machines where the load is w_k "allowed"
- The greedy algorithm places each job on the least loaded allowed machine
- It has a competitive ratio of $\lceil \log m \rceil + 1$
- No algorithm can do much better

Analysis of Greedy

- We partition the assignment of Greedy into layers
- Each layer has height $\mathrm{OPT}(\sigma)$
- Some jobs are split over two layers (not more!)
- There are n jobs
- We have

$$\mathrm{OPT}(\sigma) \geq \sum_{k=1}^{n} w_k/m$$

The layers of the greedy schedule (1)

- Let W_i be the load assigned in layer i
- Let W be the total load
- What remains after i layers have been assigned?
- Define

$$R_i = W - \sum_{\ell=1}^{i} W_\ell$$

The layers of the greedy schedule (2)

- We will show $W_i \geq R_i$ for each layer i
- Thus, in each layer Greedy assigns more than it leaves over
- If this holds, then $R_i \leq R_{i-1}/2$
- Therefore

$$R_{\lceil \log m \rceil} \leq R_0/m = W/m \leq \mathrm{OPT}(\sigma)$$

- So any load remaining after level $\lceil \log m \rceil$ will be assigned in the next level
- This shows that the maximum load is at most

$$(\lceil \log m \rceil + 1) \cdot \mathrm{OPT}(\sigma)$$

Proof of the claim ($W_i \geq R_i$)

- For layer i, let A_i be the set of machines that are allowed for one or more unfinished jobs after this layer
- The unfinished jobs contribute to R_i
- Let $N_i = |A_i|$
- We have $R_i \leq N_i \cdot \mathrm{OPT}(\sigma)$
- Let $\mathrm{FULL}_{i-1} \subset A_{i-1}$ be the set of machines in A_{i-1} that are full in level $i-1$ (get load at least $\mathrm{OPT}(\sigma)$)
- We have $W_i \geq |\mathrm{FULL}_{i-1}| \cdot \mathrm{OPT}(\sigma)$
- But we can show $N_i \leq |\mathrm{FULL}_{i-1}|$

$N_i \leq |\mathrm{FULL}_{i-1}|$

- Consider a non-full machine j in A_{i-1}
- Suppose it is allowed for some job k assigned after layer i
- Then machine j would have a load less than any machine in A_i
- So it would be assigned this job k
- Then either machine j would become full or job k would not contribute to R_i
- This shows that

$$R_i \leq N_i \cdot \mathrm{OPT}(\sigma) \leq |\mathrm{FULL}_{i-1}| \mathrm{OPT}(\sigma) \leq W_i$$