Approximation for $P||C_{max}$

Sapienza Università di Roma

Outline

Why guarantees for scheduling?

Notation

The algorithms *LIST* and *LPT*
- List scheduling
- Longest processing time first

A property of list scheduling

Performance guarantees
- The worst case performance ratio of LIST
- The worst case performance ratio of LPT

Sources

Typical mistakes

In summary

Learning objectives

At the end of this chapter you will be able to:

- execute *LIST* and *LPT* by hand and draw the resulting schedule;
- prove the guarantees $2 - \frac{1}{m}$ of *LIST* and $\frac{4}{3} - \frac{1}{3m}$ of *LPT*;
- build the families of instances that make these guarantees tight;
- explain why sorting the jobs by non-increasing processing time improves the guarantee.

Prerequisites

- From this module, the chapter «Principles of approximation algorithms»: performance ratio, relative performance guarantee, tight guarantee and worst-case performance ratio.
- From the module *Combinatorial Optimization Problems*, the chapter «Scheduling problems»: the definition and the reference instance of $P \,||\, C_{\max}$, and its elementary lower bound.
- From the module *Heuristic Algorithms*, the chapter «Constructive heuristics»: the algorithms *LIST* and *LPT* as constructive rules.

Why guarantees for scheduling?

Introduction: Why guarantees for scheduling?

- A simple greedy rule assigns the jobs one by one to the least loaded machine
- The makespan it finds depends on the **order** in which the jobs are examined
- We want to know how bad it can be, not on one instance but on *every* instance

A challenge

Seven jobs with processing times $(5, 5, 4, 4, 3, 3, 3)$ must be assigned to **three machines** so as to minimize the makespan.

Assign the jobs one by one to a machine with the smallest load.

Try the order $7, 6, 5, 4, 3, 2, 1$ and the order $1, 2, \ldots, 7$.

The answer

- The first order gives makespan 12, the second 11, and the optimal value is 9.
- How bad can the first order be on other instances?
- The positive results of this chapter bound the gap on *every* instance.

Summary

The challenge:

- the same greedy rule, two orders of the jobs: makespan 12 and 11, against the optimum 9

The goal of the chapter:

- bound the gap between the greedy makespan and the optimum on every instance
- show that the bounds cannot be improved

Notation

Introduction: Notation

- We fix the notation of the scheduling problem: machines, jobs, loads and makespan
- We recall the performance ratio of a heuristic algorithm
- We recall the elementary lower bound on the optimal value, the main tool of all the proofs

Machines, jobs and loads

- The problem of this chapter is the scheduling problem on identical parallel machines ($P \,||\, C_{\max}$): its definition and its assumptions are on its page.
- There are m identical machines and a set $\mathscr{J} = \{1, 2, \ldots, n\}$ of jobs with processing times t_j.
- A solution assigns each job $j \in \mathscr{J}$ to a machine $mach_j = h$.
- The set of jobs of machine h is J_h, and its **load** is

$$L_h = \sum_{j \in J_h} t_j.$$

- The **makespan** is $C_{\max} = \max_h L_h$.

Recall: the performance ratio

- As in the chapter «Principles of approximation algorithms», $z(I)$ is the optimal value of the instance I and $\mathscr{A}(I)$ the value of the solution returned by the algorithm $\mathscr{A}$.
- $P \,||\, C_{\max}$ is a minimization problem, hence for every instance I

$$\mathscr{A}(I) \geq z(I) \qquad \text{and} \qquad \frac{\mathscr{A}(I)}{z(I)} \geq 1.$$

- The worst case performance ratio is

$$r(\mathscr{A}) = \sup_I \frac{\mathscr{A}(I)}{z(I)}.$$

The elementary lower bound

- We use the elementary lower bound of the chapter «Scheduling problems» of the module *Combinatorial Optimization Problems*:

$$z(I) \geq \max \left\{ \max_{j \in \mathcal{J}} t_j, \quad \frac{\sum_{j \in \mathcal{J}} t_j}{m} \right\}. \tag{1}$$

- The longest job must be processed by some machine.
- The total load is shared by m machines, so some machine has at least the average load.

The reference instance

- The reference instance has

$$m = 3 \qquad \text{and} \qquad t = (5, 5, 4, 4, 3, 3, 3).$$

- The bound (1) gives

$$\max\left\{5, \frac{27}{3}\right\} = 9.$$

- The optimal value is $z(I) = 9$, with

$$J_1 = \{1, 3\}, \quad J_2 = \{2, 4\}, \quad J_3 = \{5, 6, 7\}.$$

Summary

Notation:

- m machines, jobs $\mathscr{J} = \{1, 2, \ldots, n\}$, loads L_h, makespan $C_{\max} = \max_h L_h$
- $\mathscr{A}(I) \geq z(I)$ and $r(\mathscr{A}) = \sup_I \mathscr{A}(I)/z(I)$

Lower bound:

- $z(I)$ is at least the longest job and at least the average load
- on the reference instance the bound gives $9 = z(I)$

The algorithms *LIST* and *LPT*

Introduction: The algorithms *LIST* and *LPT*

- We present the two constructive heuristics of the chapter: *LIST* and *LPT*
- *LIST* follows any list of the jobs; *LPT* sorts the jobs first
- We run both on the reference instance and draw their schedules

List scheduling

- The *List Scheduling Algorithm* (denoted by *LIST*) is a constructive heuristic for $P \,||\, C_{\max}$.
- It examines the jobs in the order of a given list λ.
- It assigns each job to a machine with the smallest current load (in case of ties, the machine with the smallest index).

Algorithm: List Scheduling (1/2)

Input : $P \,||\, C_{\max}$ instance with n jobs, times $t[1..n]$, m machines; list $\lambda[1..n]$

Output: Job-machine assignment $mach[1..n]$ and makespan $C_{\max}$

```
1 function ListScheduling(n, m, t, λ)
2 |   mach := new array[1..n]          // job-machine
  |    assignment
3 |   for h := 1 to m do
4 |   |   L[h] ← 0                    // loads of the machines
5 |   end
```

Algorithm: List Scheduling (2/2)

function *ListScheduling*(n, m, t, λ) (continued)

```
6   |  for k := 1 to n do
7   |  |  j := λ[k]                    // k-th job of the list
8   |  |  s := a machine with minimum L[s] (smallest
    |  |   index)                      // least loaded machine
9   |  |  mach[j] ← s   // assign job j to machine s
10  |  |  L[s] ← L[s] + t[j]
11  |  end
12  |  return mach, C_max = max_h L[h]
13  end
```

How *LIST* works

- *LIST* is the same algorithm of the chapter «Constructive heuristics» of the module *Heuristic Algorithms*, where it is introduced as a greedy rule.
- Here we are interested in its **worst case**: how bad can a list be?
- Each machine processes its jobs without idle times, so the job j assigned at iteration k **starts** at time S_j, equal to the load of its machine just before the assignment, and ends at $S_j + t_j$.

We run *LIST* on the reference instance
with the **shortest jobs first**.

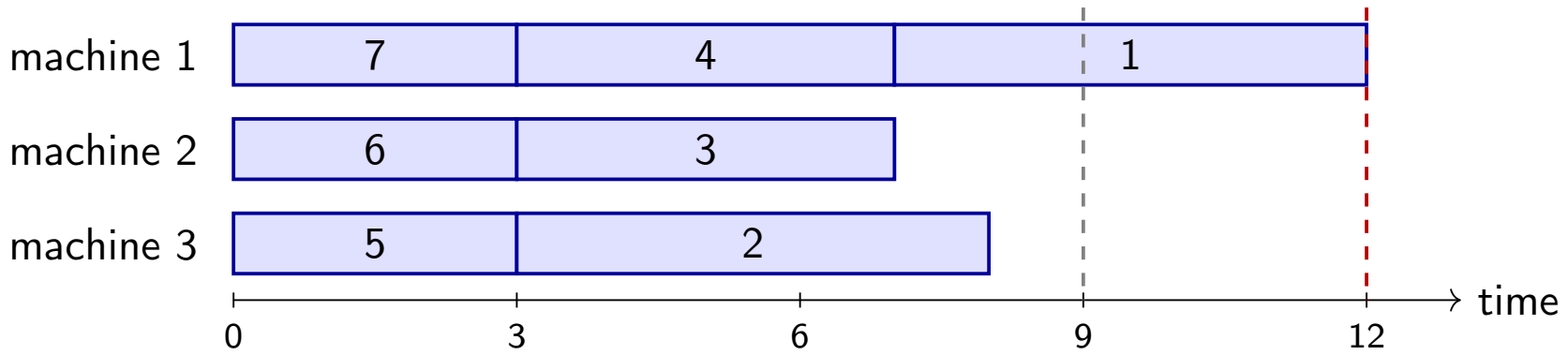
Example (1/2)

Consider the list $\lambda = (7, 6, 5, 4, 3, 2, 1)$. The loads before each assignment and the chosen machine are:

job j	t_j	loads (L_1, L_2, L_3) before	machine
7	3	$(0, 0, 0)$	1
6	3	$(3, 0, 0)$	2
5	3	$(3, 3, 0)$	3
4	4	$(3, 3, 3)$	1
3	4	$(7, 3, 3)$	2
2	5	$(7, 7, 3)$	3
1	5	$(7, 7, 8)$	1

Example (2/2)

The final loads are $(12, 7, 8)$, with $J_1 = \{7, 4, 1\}$, $J_2 = \{6, 3\}$, $J_3 = \{5, 2\}$, and $LIST(I) = 12$:



The last job, job 1, starts at $S_1 = 7$, the smallest load at that moment: all the machines are busy until time 7 at least. This is the observation on which the guarantee of *LIST* is built.

Longest processing time first

- The *Longest Processing Time first Algorithm* (denoted by *LPT*) is a constructive heuristic for $P \,||\, C_{\max}$.
- It sorts the jobs by non-increasing processing time and applies *LIST* to this list.
- Each job, from the longest to the shortest, is assigned to a least loaded machine.

Algorithm: Longest Processing Time first

Input : $P \,||\, C_{\max}$ instance with n jobs, times $t[1..n]$, m machines

Output: Job-machine assignment $mach[1..n]$ and makespan $C_{\max}$

```
1 function LPT(n, m, t)
2 |   λ := the jobs sorted by non-increasing t[j] (ties by
  |    index)                          // longest jobs first
3 |   return ListScheduling(n, m, t, λ)
4 end
```

- In the proofs it is convenient to renumber the jobs in the order of λ, so that $t_1 \geq t_2 \geq \cdots \geq t_n$ and LPT assigns the jobs in the order $1, 2, \ldots, n$.

We run *LPT* on the same reference instance:
now the **longest jobs first**.

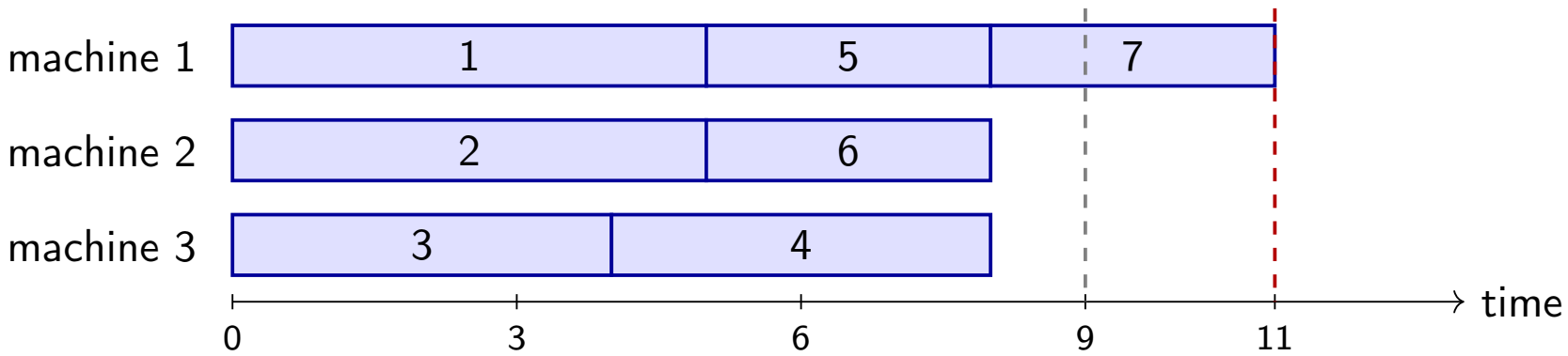
Example (1/2)

The jobs are already sorted, $\lambda = (1, 2, 3, 4, 5, 6, 7)$.

- Jobs 1, 2, 3 go to the three empty machines, with loads $(5, 5, 4)$.
- Job 4 goes to machine 3, the least loaded: $(5, 5, 8)$.
- Job 5 goes to machine 1: $(8, 5, 8)$.
- Job 6 goes to machine 2: $(8, 8, 8)$.
- Job 7 finds three machines with load 8 and goes to machine 1: $(11, 8, 8)$.

Example (2/2)

Hence $J_1 = \{1, 5, 7\}$, $J_2 = \{2, 6\}$, $J_3 = \{3, 4\}$ and $LPT(I) = 11$:



With $z(I) = 9$, the ratio is $\frac{11}{9}$: sorting improves the makespan from 12 to 11, but the last job, short as it is, unbalances three perfectly balanced machines.

Summary

$LIST$:

- each job of the list goes to a least loaded machine (smallest index in case of ties)
- job j starts at S_j, the load of its machine just before the assignment
- reference instance, shortest jobs first: $LIST(I) = 12$

LPT:

- $LIST$ applied to the jobs sorted by non-increasing processing time
- reference instance: $LPT(I) = 11$, against $z(I) = 9$

A property of list scheduling

Introduction: A property of list scheduling

- The guarantees of the next section compare the value of a heuristic solution with the lower bound (1) on $z(I)$.
- Besides the bound, they use one property of the schedules built by $LIST$: the job that ends last cannot start too late.

The start of the last job

Observation 1

Let ℓ be a job that ends last in a schedule built by LIST (with any list). Then its starting time satisfies

$$S_\ell \leq \frac{1}{m} \sum_{j \in \mathcal{J} \setminus \{\ell\}} t_j.$$

Proof

- When ℓ is assigned, S_ℓ is the smallest of the m loads, hence it is not larger than their average.
- The loads at that moment contain only jobs assigned before ℓ, so their sum is at most $\sum_{j \neq \ell} t_j$.
- Therefore

$$S_\ell \leq \frac{1}{m} \sum_{j \neq \ell} t_j.$$



Summary

The start of the last job:

- with any list, the job ℓ that ends last starts no later than the average load of the other jobs
- reason: S_ℓ is the smallest load, hence at most the average load

What comes next:

- this property and the lower bound (1) give the guarantees of *LIST* and *LPT*

Performance guarantees

Introduction: Performance guarantees

- We prove the guarantee $2 - \frac{1}{m}$ of $LIST$, for every list
- We prove the better guarantee $\frac{4}{3} - \frac{1}{3m}$ of LPT
- For each guarantee we build a family of instances on which it is reached: both guarantees are **tight**

The guarantee of *LIST*

Proposition 1

For every $P \,||\, C_{\max}$ instance I and every list λ,

$$\frac{LIST(I)}{z(I)} \leq 2 - \frac{1}{m}.$$

- Idea: bound the start of the last job with the observation of the previous section, and its length with the lower bound (1).

Proof

Let ℓ be a job that ends last, so that $LIST(I) = S_\ell + t_\ell$. By the observation on the start of the last job,

$$LIST(I) \leq \frac{1}{m} \sum_{j \neq \ell} t_j + t_\ell$$

$$= \underbrace{\frac{1}{m} \sum_{j \in \mathscr{J}} t_j}_{\leq z(I)} + \left(1 - \frac{1}{m}\right) \underbrace{t_\ell}_{\leq z(I)}$$

$$\leq \left(2 - \frac{1}{m}\right) z(I),$$

where both bounds come from (1). Dividing by $z(I)$ gives the claim. □

Can a list be so bad that the ratio reaches $2 - \frac{1}{m}$?

The guarantee of *LIST* is tight

Proposition 2

For every $m \geq 2$ there is a $P||C_{\max}$ instance I_m with m machines and a list λ such that

$$\frac{LIST(I_m)}{z(I_m)} = 2 - \frac{1}{m}.$$

Proof (1/3)

- I_m has $n = m(m-1)+1$ jobs:
  - jobs $1, \ldots, m(m-1)$ with $t_j = 1$;
  - job n with $t_n = m$.
- The list is $\lambda = (1, 2, \ldots, n)$.
- We have $n > m$ for $m \geq 2$.

Proof (2/3)

- The unit jobs are assigned in rounds of m: in each round all the machines have the same load, the jobs go to machines $1, 2, \ldots, m$ in this order and every load grows by 1.
- After the $m-1$ rounds all the loads are $m-1$, and job n starts at $m-1$ on machine 1:

$$LIST(I_m) = (m-1) + m = 2m - 1.$$

Proof (3/3)

- Putting job n alone on machine 1 and the $m(m-1)$ unit jobs m per machine on machines $2, \ldots, m$ gives makespan m.
- The bound (1) gives $z(I_m) \geq t_n = m$.
- Hence $z(I_m) = m$ and

$$\frac{LIST(I_m)}{z(I_m)} = \frac{2m-1}{m} = 2 - \frac{1}{m}.$$



The worst case performance ratio of *LIST*

- From the two propositions above it follows that *LIST* is a $(2 - \frac{1}{m})$-approximation algorithm, whatever the list.
- The worst case performance ratio of *LIST* on m machines is

$$r(LIST) = 2 - \frac{1}{m}$$

Remarks on *LIST*

- For $m = 3$ the family has $t = (1, 1, 1, 1, 1, 1, 3)$, $LIST(I_3) = 5$ and $z(I_3) = 3$.
- The bad job is the long one, put last: *LPT* would put it first and find the optimum.
- On the example of *LIST* with the shortest jobs first,

$$LIST(I) = 12 \leq \frac{5}{3} \cdot 9 = 15.$$

- The bound and the family are due to Graham (1966).

Sorting the jobs puts the long jobs first.

How much does this improve the guarantee?

The guarantee of *LPT*

Proposition 3

For every $P \,||\, C_{\max}$ *instance* I,

$$\frac{LPT(I)}{z(I)} \leq \frac{4}{3} - \frac{1}{3m}.$$

- Idea: keep only the jobs up to the last one, then distinguish a short last job (the argument of *LIST*) from a long last job (at most two jobs per machine).

Proof (1/5)

- Number the jobs so that $t_1 \geq t_2 \geq \cdots \geq t_n$, the order in which LPT assigns them.
- Let ℓ be a job that ends last, so that $LPT(I) = S_\ell + t_\ell$.
- The jobs after ℓ play no role: let I' be the instance with the jobs $1, \ldots, \ell$ only.
- LPT assigns them on I' exactly as on I, so $LPT(I') = LPT(I)$.
- $z(I') \leq z(I)$, since a schedule of I restricted to the jobs of I' is a schedule of I'.

Proof (2/5)

- It is therefore enough to prove $LPT(I') \leq \left(\frac{4}{3} - \frac{1}{3m}\right) z(I')$.
- In I' the job ℓ is the shortest one. We distinguish two cases.

Case 1: $t_\ell \leq \frac{z(I')}{3}$. As in the proof for $LIST$, with the observation on the start of the last job applied to I',

$$\begin{aligned} LPT(I') &\leq \frac{1}{m}\sum_{j=1}^{\ell} t_j + \left(1 - \frac{1}{m}\right) t_\ell \\ &\leq z(I') + \left(1 - \frac{1}{m}\right)\frac{z(I')}{3} \\ &= \left(\frac{4}{3} - \frac{1}{3m}\right) z(I'). \end{aligned}$$

Proof (3/5)

Case 2: $t_\ell > \frac{z(I')}{3}$.

- Every job of I' is longer than $\frac{z(I')}{3}$, so in an optimal schedule of I' no machine has three jobs, and $\ell \leq 2m$.
- If $\ell \leq m$, job ℓ starts on an empty machine and $LPT(I') = t_\ell \leq z(I')$.
- Otherwise $\ell = m + r$ with $1 \leq r \leq m$; let $k := m - r + 1$, so that $1 \leq k \leq m$.
- We show that

$$z(I') \geq t_k + t_\ell \geq LPT(I').$$

Proof (4/5)

Lower bound. Consider an optimal schedule of I'.

- If two of the jobs $1, \ldots, k$ share a machine, its load is at least $2t_k \geq t_k + t_\ell$.
- Otherwise they lie on k different machines; the other $m - k = r - 1$ machines receive at most $2(r-1)$ jobs, while the jobs $k+1, \ldots, \ell$ are $\ell - k = 2r - 1$.
- So one of them shares a machine with one of the jobs $1, \ldots, k$, and that load is at least $t_k + t_\ell$.
- In both cases $z(I') \geq t_k + t_\ell$.

Proof (5/5)

Upper bound.

- The first m jobs go to m different machines, since an empty machine has load 0 and every job has positive length.
- Before job ℓ, the m machines hold $\ell - 1 = m + r - 1$ jobs, each machine at least one, so at most $r - 1$ machines hold two jobs or more and at least $m - r + 1 = k$ machines hold exactly one job, one of the jobs $1, \ldots, m$.
- These are k different jobs among $1, \ldots, m$, so the largest index among them is at least k, and its length is at most t_k.
- Hence $S_\ell \leq t_k$, and $LPT(I') = S_\ell + t_\ell \leq t_k + t_\ell$.

Therefore $LPT(I') \leq z(I')$ in Case 2, which is even better than the claim. □

Is the guarantee of *LPT* tight?

We build a family of instances that reaches it.

The guarantee of *LPT* is tight

Proposition 4

For every $m \geq 2$ there is a $P||C_{\max}$ instance I_m with m machines such that

$$\frac{LPT(I_m)}{z(I_m)} = \frac{4}{3} - \frac{1}{3m}.$$

Proof (1/5)

- I_m has $n = 2m + 1$ jobs, already sorted: two jobs of each length $2m - 1, 2m - 2, \ldots, m + 1$, and three jobs of length m,

$$t = (2m-1,\ 2m-1,\ 2m-2,\ 2m-2,\ \ldots,\ m+1,\ m+1,\ m,\ m,\ m).$$

- For $j \leq 2m$ the length is

$$t_j = 2m - \lceil j/2 \rceil,$$

and $t_{2m+1} = m$.

Proof (2/5)

Optimum.

- The total length is

$$2 \sum_{v=m+1}^{2m-1} v + 3m = 3m^2,$$

  so (1) gives $z(I_m) \geq 3m$.
- The schedule that pairs a job of length v with a job of length $3m - v$, for $v = m + 1, \ldots, 2m - 1$ (both lengths lie in $\{m + 1, \ldots, 2m - 1\}$, and each length is used twice), fills $m - 1$ machines with load $3m$.
- It puts the three jobs of length m on the last machine, again with load $3m$.
- Hence $z(I_m) = 3m$.

Proof (3/5)

LPT.

- Jobs $1, \ldots, m$ go to machines $1, \ldots, m$.
- Then job $m+1$ goes to a machine with the smallest single job, t_m, and its load becomes $t_m + t_{m+1}$.
- In general, we claim that job $m+q$, for $q = 1, \ldots, m$, goes to a machine whose only job has length t_{m+1-q}, and the load becomes

$$t_{m+1-q} + t_{m+q} = 3m - 1.$$

Proof (4/5)

The identity. Let $j = m + 1 - q$, so that $2m + 1 - j = m + q$.

- If $j = 2i - 1$ is odd, then

$$\lceil j/2 \rceil = i \qquad \text{and} \qquad \lceil (2m + 1 - j)/2 \rceil = m - i + 1.$$

- If $j = 2i$ is even, the two ceilings are again i and $m - i + 1$.
- In both cases

$$t_j + t_{2m+1-j} = (2m - i) + (m + i - 1) = 3m - 1.$$

Proof (5/5)

The claim. It follows by induction.

- A machine with two jobs has load $3m - 1 > 2m - 1 \geq$ any single job, so it is never the least loaded while machines with one job remain.
- Among the machines with one job, the least loaded holds the shortest remaining job, of length t_{m+1-q} (with equal lengths the loads are the same).
- After $2m$ jobs all the loads are $3m - 1$, and the last job, of length m, gives

$$LPT(I_m) = 3m - 1 + m = 4m - 1,$$

$$\frac{LPT(I_m)}{z(I_m)} = \frac{4m-1}{3m} = \frac{4}{3} - \frac{1}{3m}.$$

□

The worst case performance ratio of *LPT*

- From the two propositions on *LPT* it follows that *LPT* is a $\left(\frac{4}{3} - \frac{1}{3m}\right)$-approximation algorithm.
- Its worst case performance ratio on m machines is

$$r(LPT) = \frac{4}{3} - \frac{1}{3m}$$

- Sorting the list lowers the guarantee of *LIST* from $2 - \frac{1}{m}$ to $\frac{4}{3} - \frac{1}{3m}$.

Remarks on *LPT*

- For $m = 3$ the family is $t = (5, 5, 4, 4, 3, 3, 3)$, the reference instance: the loads after six jobs are $(8, 8, 8) = 3m - 1$, the seventh job gives $LPT(I) = 11 = 4m - 1$, and $z(I) = 9 = 3m$ (the example of *LPT* above).
- The reference instance is a worst case for *LPT*: $\frac{11}{9} = \frac{4}{3} - \frac{1}{9}$.
- On the tight instances the last job has $t_\ell = m = \frac{z(I_m)}{3}$: they lie exactly on the border between the two cases of the proof, and every inequality of Case 1 holds with equality.
- The bound and the family are due to Graham (1969).

Summary (1/2)

LIST:

- $LIST(I) \leq \left(2 - \frac{1}{m}\right) z(I)$ for every list: last job's start plus its length, both bounded by $z(I)$
- tight: $m(m-1)$ unit jobs followed by one job of length m

LPT:

- $LPT(I) \leq \left(\frac{4}{3} - \frac{1}{3m}\right) z(I)$: Case 1 short last job, Case 2 at most two jobs per machine
- tight: the family of Graham, the reference instance for $m = 3$

Summary (2/2)

Worst case performance ratios:

- $r(LIST) = 2 - \frac{1}{m}$
- $r(LPT) = \frac{4}{3} - \frac{1}{3m}$

The role of sorting:

- sorting the list by non-increasing processing time lowers the guarantee from $2 - \frac{1}{m}$ to $\frac{4}{3} - \frac{1}{3m}$

Sources

Introduction: Sources

- We list the two papers of Graham where the guarantees of this chapter were proved
- The first one is about *LIST*, the second one about *LPT*

Sources

- R.L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal* 45(9):1563–1581, 1966.
- R.L. Graham, Bounds on multiprocessing timing anomalies, *SIAM Journal on Applied Mathematics* 17(2):416–429, 1969.

Summary

Sources:

- Graham (1966): the guarantee $2 - \frac{1}{m}$ of $LIST$ and its tight family
- Graham (1969): the guarantee $\frac{4}{3} - \frac{1}{3m}$ of LPT and its tight family

Typical mistakes

Introduction: Typical mistakes

- We collect the mistakes that are made most often with *LIST*, *LPT* and their guarantees
- Check them before solving the exercises

Typical mistakes (1/2)

Observation 2: Using the guarantee of *LPT* for an unsorted list

LIST has the guarantee $2-\frac{1}{m}$ *for every list; the better guarantee* $\frac{4}{3}-\frac{1}{3m}$ *needs the jobs sorted by non-increasing processing time.*

Observation 3: Computing the ratio with the lower bound

The bound (1) *can be smaller than* $z(I)$*: the ratio* $\mathscr{A}(I)/\mathrm{LB}$ *is only an upper bound on the performance ratio* $\mathscr{A}(I)/z(I)$*, and the two coincide only when the bound is tight.*

Typical mistakes (2/2)

Observation 4: Breaking ties with another rule

LIST assigns each job to a least loaded machine of smallest index; another tie rule can give a different schedule, and the solutions of the exercises follow the rule of the List Scheduling algorithm.

Summary

Guarantees:

- $\frac{4}{3} - \frac{1}{3m}$ only for the sorted list; $2 - \frac{1}{m}$ for any list
- $\mathscr{A}(I)/\mathrm{LB}$ only bounds the performance ratio from above

Ties:

- a least loaded machine of smallest index, as in the List Scheduling algorithm

In summary

Introduction: In summary

- We collect the key ideas of the chapter on the approximation of $P \,||\, C_{\max}$
- The dashboard collects the bounds on $z(I)$ found so far for the reference instance

Summary (1/2)

- The guarantees of *LIST* and *LPT* are proved with the load bound (1) on $z(I)$ and with the observation on the start of the last job: the job that ends last starts no later than the average load of the other jobs.
- Both ratios are tight, with Graham's families of instances; for $m = 3$ the reference instance is the tight instance of *LPT*.
- Sorting the list lowers the guarantee from $2 - \frac{1}{m}$ to $\frac{4}{3} - \frac{1}{3m}$.

Summary (2/2)

problem	algorithm	worst case ratio
$P \mid\mid C_{\max}$	*LIST*	$2 - \frac{1}{m}$ (tight)
$P \mid\mid C_{\max}$	*LPT*	$\frac{4}{3} - \frac{1}{3m}$ (tight)

Dashboard of the reference instance. The bounds on $z(I)$ found so far:

Instance	$P \mid\mid C_{\max}$, $m = 3$, $t = (5, 5, 4, 4, 3, 3, 3)$
Lower bound	9, load bound
Heuristic values	*LIST* 12 (list $7, 6, \ldots, 1$), *LPT* 11
Optimal value	9