Parallel Machine Problems

1. Minimising $C_{\max}$

For the problem with parallel machines, the following two lower bounds of the makespan are often used.

Job-based bound: $C_{\max} \geq p$, where $p = \max\{p_1, p_2, \cdots, p_n\}$ (1)

(makespan cannot be smaller than the time required to complete one job)

Machine-based bound: $C_{\max} \geq \sum_{j=1}^{n} p_j / m$ (2)

(*makespan cannot be smaller than the average machine load*)

Both lower bounds hold for any schedule, preemptive or non-preemptive, optimal or non-optimal.

Observe that (1) and (2) can be replaced by

$$C_{\max} \geq \max\left\{p, \sum_{j=1}^{n} p_j / m\right\} \qquad (3)$$

1.1 Preemptive case (problem $P|pmtn|C_{\max}$)

For the preemptive problem $P|pmtn|C_{\max}$, lower bound (3) can be achieved, i.e., for the makespan of the optimal schedule condition (3) holds as equality:

$$\boxed{C_{\max}^{OPT} = \max\left\{p, \sum_{j=1}^{n} p_j / m\right\}}$$

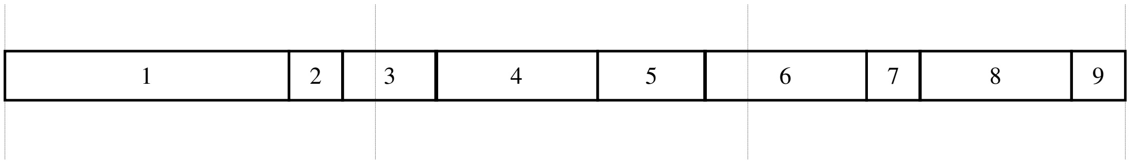
The algorithm below constructs such a schedule.

<u>Wrap-around algorithm:</u>

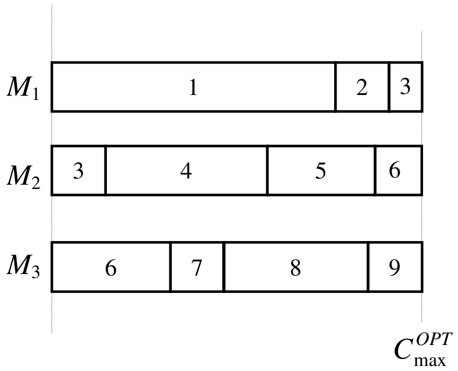
1. Calculate the optimal makespan value $C_{\max}^{OPT} = \max\left\{p, \sum_{j=1}^{n} p_j / m\right\}$.
2. Construct a single-machine nonpreemptive schedule by assigning n jobs to a single machine in an arbitrary order starting with the longest job.
3. Cut this single-machine schedule into m parts of length $C_{\max}^{OPT}$ (the last part may be shorter).
   - Take the processing sequence of the first part as the schedule for machine M_1.
   - Take the processing sequence of the second part as the schedule for machine M_2
   - Continue with the remaining machines in a similar way.

Example 1: $C_{\max}^{OPT} = \sum_{j=1}^{n} p_j / m$

Single machine schedule



Parallel machine schedule

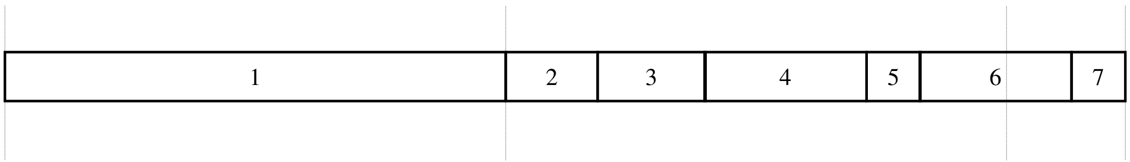


Jobs 3 and 6 are processed with preemption.

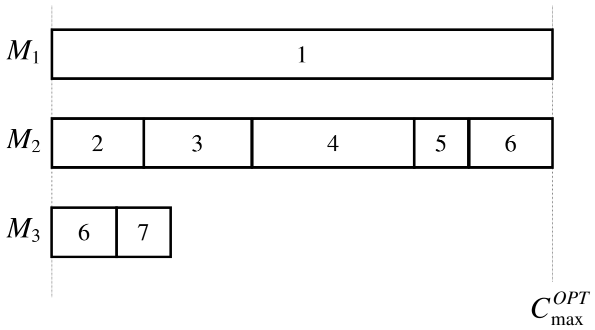
The resulting schedule is ***feasible***, i.e., no one job is processed by two machines at a time.

Example 2: $C_{\max}^{OPT} = p$

Single machine schedule



Parallel machine schedule



Job 6 is processed with preemption.

The resulting schedule is ***feasible***, i.e., no one job is processed by two machines at a time.

1.2 Non-preemptive case (problem $P||C_{max}$)

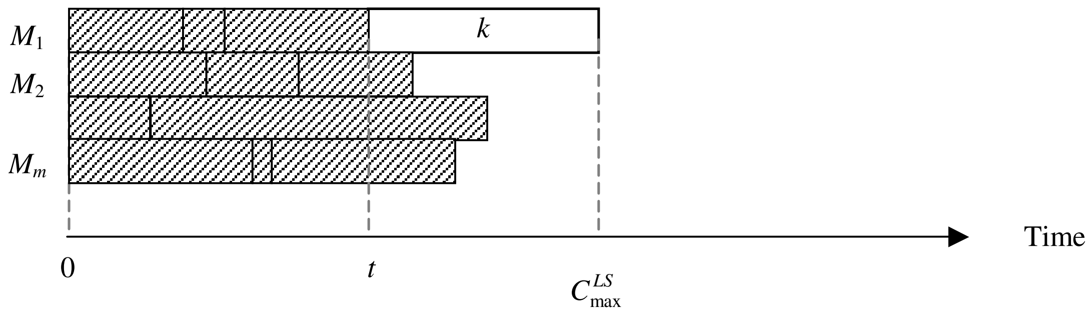
The nonpreemptive problem $P||C_{max}$ is NP-hard. Hence it is unlikely that the optimal schedule can be found in polynomial time.

The following simple algorithm finds an approximate solution in $O(n)$ time**.**

Algorithm List Scheduling

- place jobs into a list (in an arbitrary order)
- schedule the first available job from the list of unscheduled jobs whenever a machine becomes idle.

The structure of schedule S^{LS} can be illustrated by the following figure.



For P|| C_{max}, list scheduling is a 2-approximation algorithm.

We show that for the schedule S^{LS} constructed by list scheduling the following inequality holds:

$$\frac{C_{max}^{LS}}{C_{max}^{OPT}} \leq 2.$$

Let job k be the last job in the list and t be its start time.
No machine is idle before the start time of job k. Due to this fact

$$mt \leq \sum_{j=1}^{n} p_j .$$

Since k is the last job in the schedule,

$$C_{max}^{LS} = t + p_k \leq \frac{1}{m}\sum_{j=1}^{n} p_j + p_k .$$

Due to lower bound (2), $\frac{1}{m}\sum_{j=1}^{n} p_j \leq C_{max}^{OPT}$,

and due to lower bound (1), $p_k \leq p \leq C_{max}^{OPT}$.

We conclude that $C_{max}^{LS} \leq 2C_{max}^{OPT}$, i.e., list scheduling is a 2-approximation algorithm.

In fact a stronger result can be proved for list scheduling: the algorithm has a worst-case ratio of 2-1/m as shown by Graham in 1965 in the first paper on the worst-case analysis of scheduling heuristics.

*If in the List Scheduling Algorithm the job list is sorted in order of nonincreasing processing times, then this algorithm is known as **LPT** (longest processing time first).*
It can be proved, that for the LPT-algorithm, the worst-case ratio is $4/3 - 1/(3m)$.

2. Minimising ΣC_j and $\Sigma w_j C_j$

We demonstrated earlier that SPT rule finds an optimal schedule for a single machine problem $1||\Sigma C_j$. It can be proved that it is also optimal for the parallel machine problem $P||\Sigma C_j$.

As far as $\Sigma w_j C_j$ objective is concerned, problem $P||\Sigma w_j C_j$ is NP-hard.
WSPT-rule turns out to be a good approximation algorithm. It can be proved that its worst-case ratio is 1.21 (i.e., WSPT is 1.21-approximation algorithm for $P||\Sigma w_j C_j$).