Contents

1. Lecture 4

Definition **Scheduling** concerns allocating resources over time, to a set of tasks/activities/jobs.

The resources and tasks in an organization can take many different forms.

The resources may be machines, runways at an airport, people, space, and so on.
The tasks/jobs may be production,take-offs and landings at an airport, jobs, take-offs and landings at an airport, and so on.
Each task may have a certain priority level, an earliest possible starting time and a due date.

The number of jobs is denoted by n and the number of machines by m. Usually, the subscript j refers to a job while the subscript i refers to a machine.

If a job requires a number of processing steps / operations, then the pair (i, j) refers to the processing step or operation of job j on machine i.

Processing time p_j	p_{ij} represents the **processing time** of job j on machine i. The subscript i is omitted if the processing time of job j does not depend on the machine or if job j is only to be processed on one given machine.
Due date d_j	The **due date** d_j of job j represents the date the job is promised to the customer.
Release date r_j	r_j is the earliest time at which job j can start its processing.
Completion time c_j	completion time of a job
Weight w_j	The **weight** w_j of job j is basically a priority factor, denoting the importance of job j relative to the other jobs in the system.

A **scheduling problem** is described by a triplet $\alpha \mid \beta \mid \gamma$.

- The α field describes **the machine environment** and contains just one entry.
- The β field provides details of **job characteristics** and constraints and may contain no entry at all, a single entry, or multiple entries.
- The γ field describes the **objective** criterion to be minimized/maximized and often contains a single entry.

Example: $1 \mid r_j \mid \sum_j Cj$

- Single machine.
- Jobs have release dates.
- Objective is minimizing the sum of the completion times.

The possible machine environments specified in the α field are:

- **A single machine** ($\alpha = 1$) is the simplest of all possible machine environments and is a special case of all other more complicated machine environments.

- **Identical machines in parallel** ($\alpha = Pm$), there are m identical machines in parallel. Job j requires a single operation and may be processed on any one of the m machines or on any one that belongs to a given subset. p_j is the process time of job j.

- **Machines in parallel with different speeds** ($\alpha = Qm$), there are m machines in parallel with different speeds. The speed of machine i is denoted by s_i. The time p_{ij} that job j spends on machine i is equal to p_j/s_i (assuming job j receives all its processing from machine i).
- **Unrelated machines in parallel** ($\alpha = Rm$), there are m different machines in parallel. Machine i can process job j at speed s_{ij} . The time p_{ij} that job j spends on machine i is equal to p_j/v_{ij} (assuming job j receives all its processing from machine i).

The processing restrictions and constraints specified in the β field may include multiple entries, they are:

- **Release dates**, if the symbol r_j appears in the β field, then job j cannot start its processing before its release date. If rj does not appear in the β field, the processing of job j may start at any time.

- **Due date**, the job j should finish before its due date d_j.

- **Preemptions** (prmp), they imply that it is not necessary to keep a job on a machine, once started, until its completion. The scheduler is allowed to interrupt the processing of a job (preempt) at any point in time and put a different job on the machine instead. When preemptions are allowed prmp is included in the β field; when prmp is not included, preemptions are not allowed
- **Precedence constraints** (prec), they may appear in a single machine or in a parallel machine environment, requiring that one or more jobs may have to be completed before another job is allowed to start its processing. If no prec appears in the β field, the jobs are not subject to precedence constraints.
- **Unit processing times** ($p_j = 1$), each job (operation) has unit processing times.

The objective to be minimized is always a function of the completion times of the jobs, which, of course, depend on the schedule. The completion time of the operation of job j on machine i is denoted by C_{ij}. The time job j exits the system (that is, its completion time on the last machine on which it requires processing) is denoted by C_j. Examples of possible objective functions to be minimized are:

- The **makespan** $C_{\max}$ defined as $\max(C_1, \ldots, C_n)$, is equivalent to the completion time of the last job to leave the system. A minimum makespan usually implies a good utilization of the machine(s).

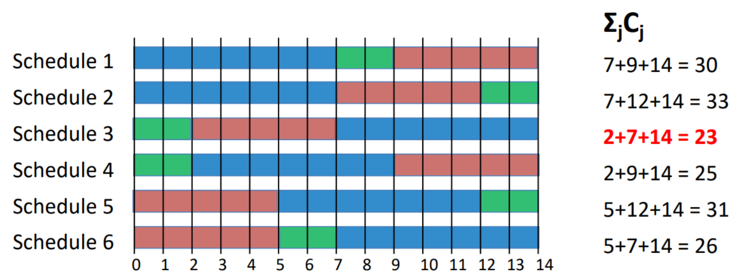
- The **maximum lateness** $L_{\max}$ is defined as $\max(L_1, \ldots, L_n)$. It measures the worst violation of the due dates.
  The lateness of job j is defined as $L_j = C_j - d_j$ (positive when job j is completed late and negative when it is completed early)

- **Total completion time** : $\sum_j C_j$
- **Total weighted completion time** $\sum_j C_j \cdot w_j$, the sum of the weighted completion times of the n jobs gives an indication of the total holding or inventory costs incurred by the schedule.

1.0.1 Example 1

$$1 \mid \cdot \mid \sum_j c_j$$

jobs	1	2	3
length p_j	7	2	5



Theorem Shortest Processing Time (SPT) rule is optimal.

Proof. Let the jobs be $j_1, \ldots, j_n$ with processing time $(p_1, \ldots, p_n)$ with $0 \leq p_1 \leq \cdots \leq p_n$

$$\sum_{k=1}^{n} c_k = n \cdot p_1 + (n-1)p_2 + \cdots + p_n$$

We want to prove that $n \cdot p_1 + (n-1)p_2 + \cdots + p_n$ is minimum.

Take any other sequence by swapping C_i and C_j with $1 \leq i < n$ and $i < j \leq n$ Assume our sequence is not minimal. Therefore there are subscripts i and j such that

$$\begin{aligned} c_1 + \cdots + c_i + \cdots + c_j + \cdots + c_n &\geq c_1 + \cdots + c_j + \cdots + c_{i+j} + \cdots + c_n \\ c_i + \cdots + c_j + \cdots + c_n &\geq c_j + \cdots + c_{j+i} + \cdots + c_{j+n} \\ -c_{j-i} - \cdots - c_i - \cdots - c_j &\geq 0 \\ -(c_{j-i} + \cdots + c_i + \cdots + c_j) &\geq 0 \end{aligned}$$

It's clear that a sum of positive elements cannot be negative therefore we have a contradiction. □

Theorem In $1 \mid p_j = 1 \mid \sum_j C_j \cdot w_j$, decreasing order of weights is optimal.

Proof. Let the jobs be $j_1, \ldots, j_n$ with weight $(w_1, \ldots, w_n)$ with $w_n \geq \cdots \geq w_1$

$$\sum_{k=1}^{n} c_k w_k = n \cdot w_1 + \cdots + w_n$$

We want to prove that expression is minimal (same proof) □

jobs j	1	2	3
weight w_j	10	5	2
length p_j	7	2	6

Schedule 1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	7*10+ 9*5 + 15*2 = 145
Schedule 2	1	1	1	1	1	1	1	3	3	2	2	2	2	2	7*10+ 13*2 + 15*5 = 171
Schedule 3	2	2	2	2	2	1	1	1	1	1	1	1	3	3	2*5 + 9*10+ 15*2 = 130
Schedule 4	2	2	2	2	2	3	3	1	1	1	1	1	1	1	2*5+8*6 +15*10=208
Schedule 5	3	3	1	1	1	1	1	1	1	2	2	2	2	2	2*6+9*10 +15*2=132
Schedule 6	3	3	2	2	2	2	2	1	1	1	1	1	1	1	2*6+8*5+15*10=202

How to order? By weight? By length?
Notice that $\frac{5}{2} > \frac{10}{7} > \frac{2}{6}$ therefore we should order according to the ratio and we find again that (2,1,3) is the best order.

Theorem In $1 \mid \cdot \mid \sum_j C_j \cdot w_j$ Weighted Shortest Processing Time (=Smith's rule) is optimal.

Proof. Let the jobs be $j_1, \dots, j_n$ with weight $(w_1, \dots, w_n)$ and $p_1, \dots, p_n$ such that $\frac{w_1}{p_1} \geq \cdots \geq \frac{w_n}{p_n}$

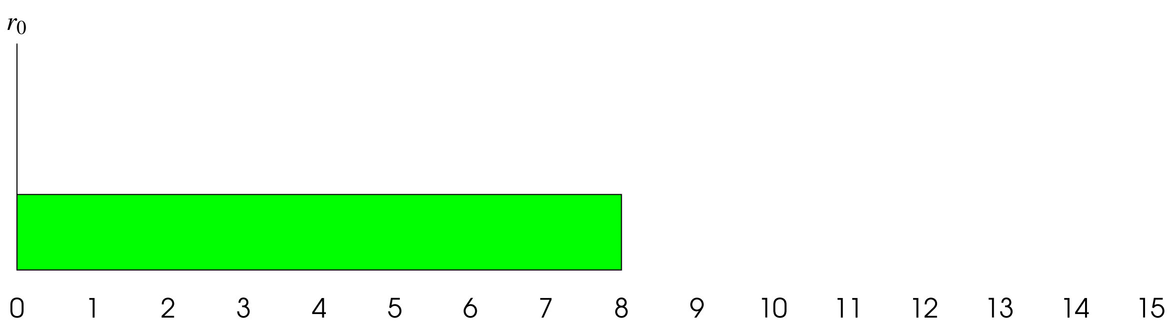
$$\sum_{k=1}^{n} c_k w_k = n \cdot w_1 + \cdots + w_n$$

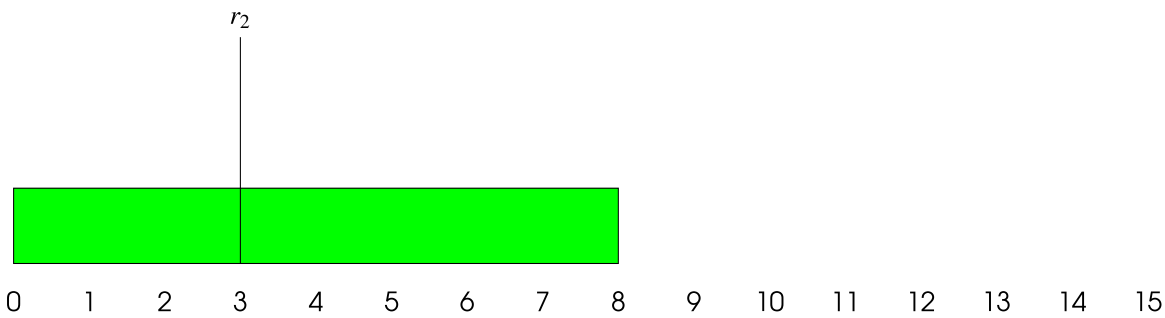
We want to prove that expression is minimal (same proof) □

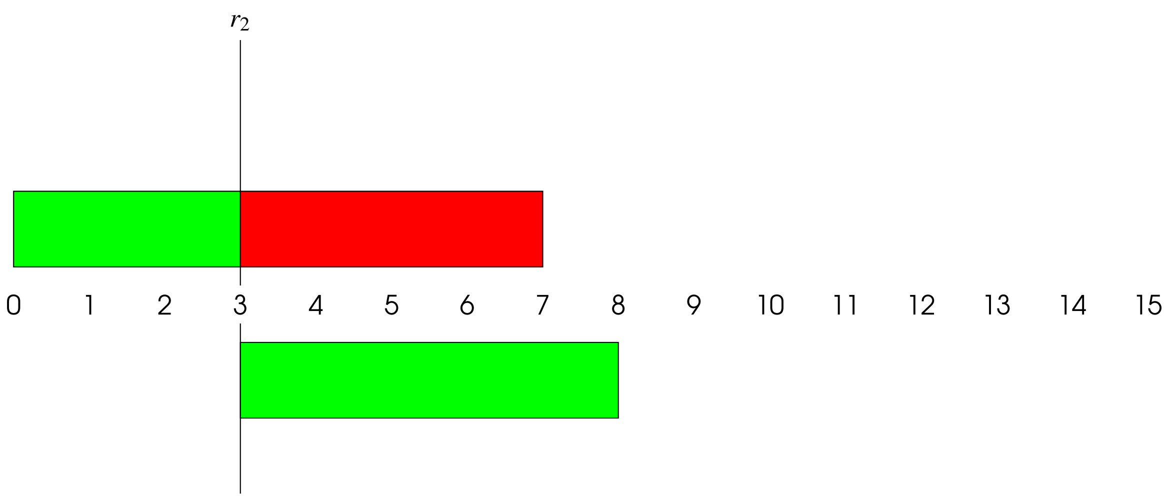
jobs j	1	2	3	4
release time r_j	0	3	4	7
length p_j	8	4	1	2

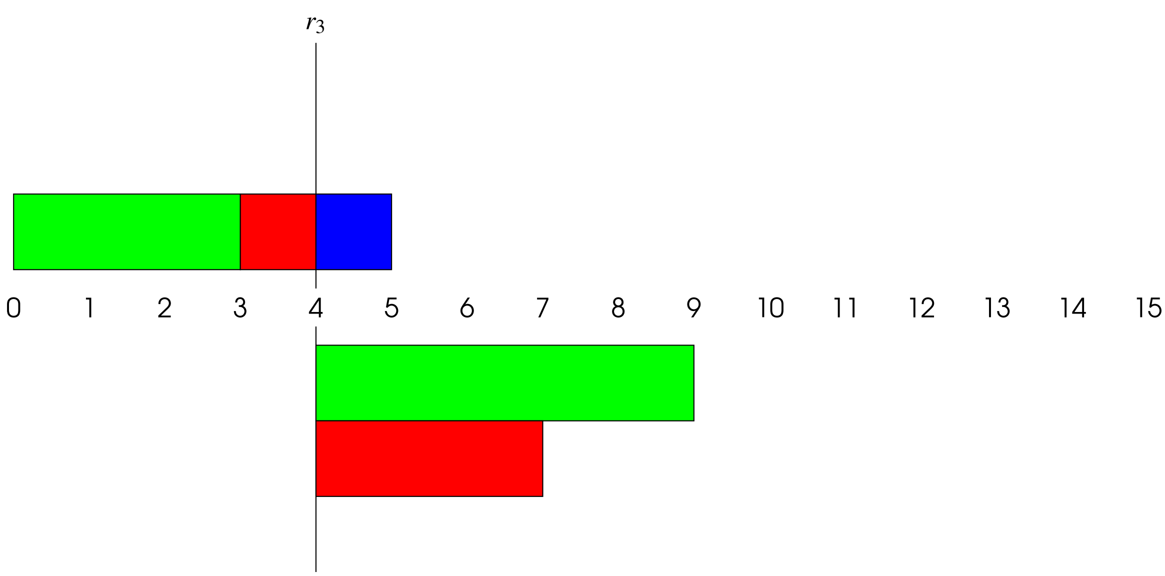
How to order in $1 \mid r_j, prmp \mid \sum_j C_j$?

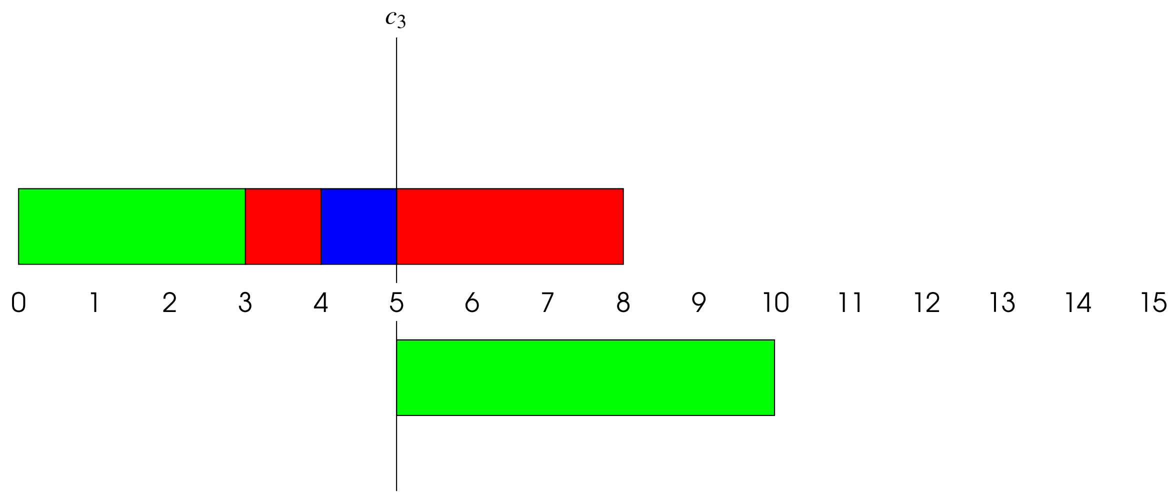
The optimal way is to process the job with the Smallest Remaining Processing Time at any moment:



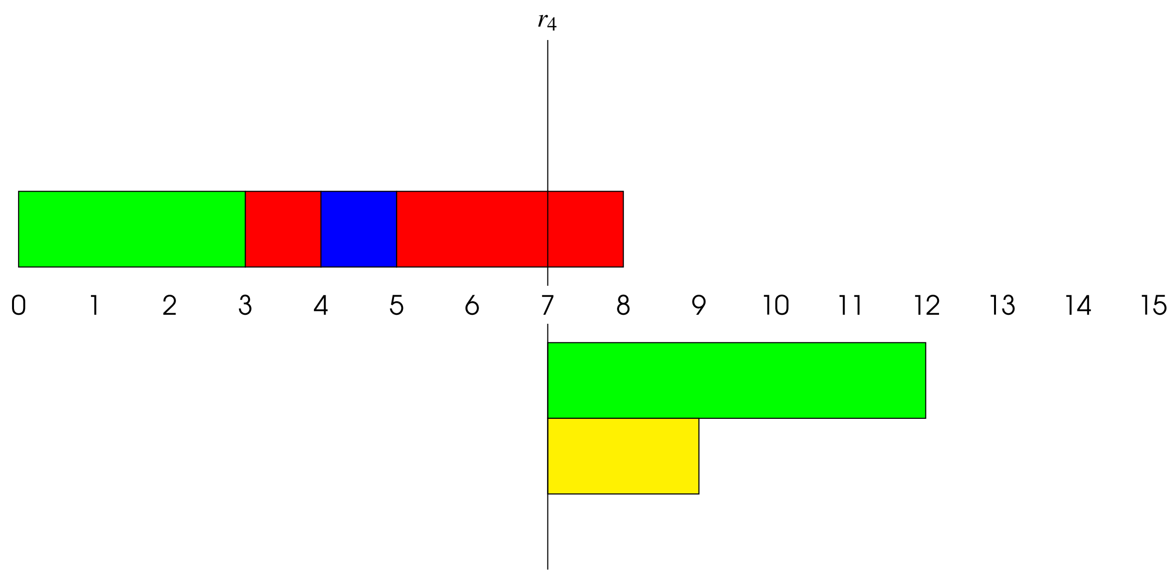

r_2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


r_2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

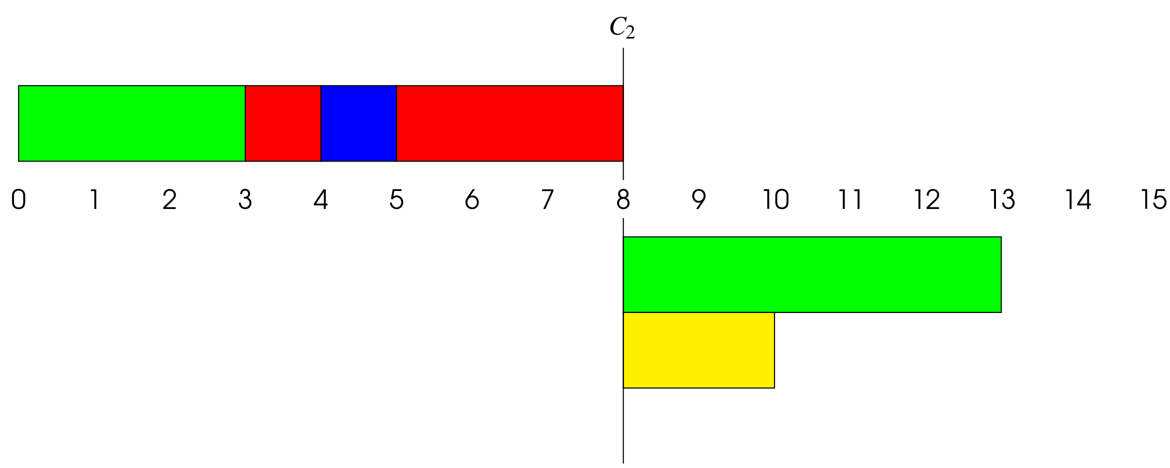

r_3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



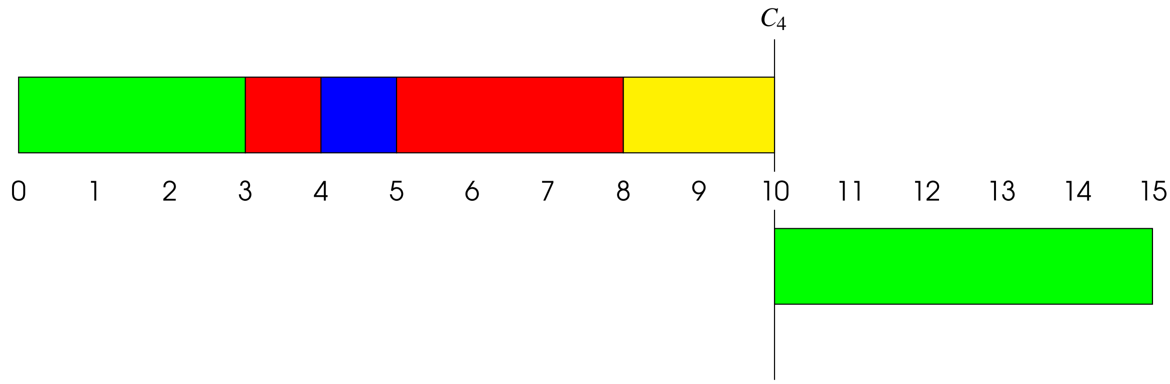
c_3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

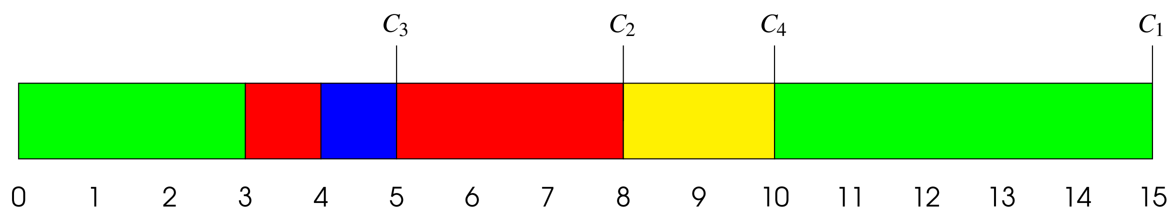


r_4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



C_2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15





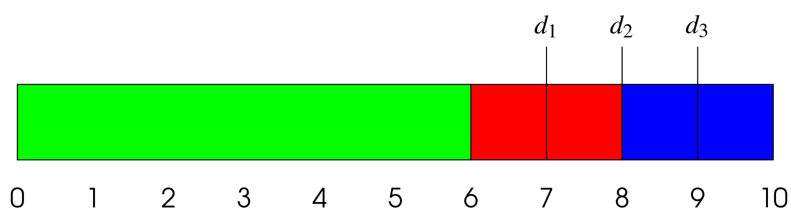
Theorem SRPT is optimal

Proof. □

Now let's consider SRPT on m parallel machines: At any moment in time, process the m jobs with smallest remaining processing time (or all jobs if there are less than m jobs available at that time).

In tutorial we will show that in $P \mid r_j, pmtn \mid \sum_j c_j$, SRPT is not optimal on parallel machines

jobs j	1	2	3
due date d_j	7	8	9
length p_j	6	2	2



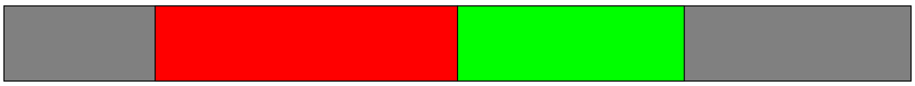
$$L_{\max} = \max(L_1, L_2, L3) = \max((-1, 0, 1) = 1$$

Theorem Earliest Due Date is optimal in $1 \mid d_j \mid L_{\max}$

Proof. We assume that there exists an optimal schedule S which does not follow the EDF rule.

We consider the first pair of consecutive tasks k, l processed in this order and such that $d_k > d_l$:

We call S' the schedule obtained by exchanging k and l in S:



In the schedule S, we have $L_k = C_k - d_k$ and $L_l = C_l - d_l = C_k + p_l - d_l$.
In the schedule S', we have

$$L'_l = C_l - p_l - d_l < C_l - d_l = L_l$$

$$L'_k = C_k - d_k + p_l = L_l + d_l - d_k < L_l$$

$$L'_{\max} = \max(L'_k, L'_l) < L_l \leq \max(L_k, L_k) = L_{\max}$$

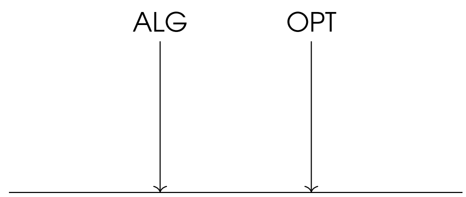
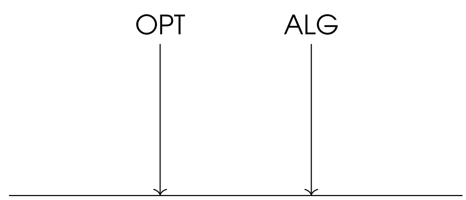
Our assumption that there is a more optimal schedule which does not follow the EDF is wrong therefore EDF is optimal. □

We can resume all case via a table:

$1 \mid \cdot \mid \sum_j c_j$	SPT: Shortest processing time, ordering by increasing length
$1 \mid p_j = 1 \mid \sum_j c_j \cdot w_j$	Decreasing ordering of weights
$1 \mid \cdot \mid \sum_j c_j \cdot w_j$	WSPT: Weighted shortest processing time, ordering by w_j/p_j is optimal
$1 \mid r_j, pmtn \mid \sum_j c_j \cdot w_j$	SRPT: Smallest remaining processing time
$1 \mid d_j \mid L_{\max}$	Earliest due date

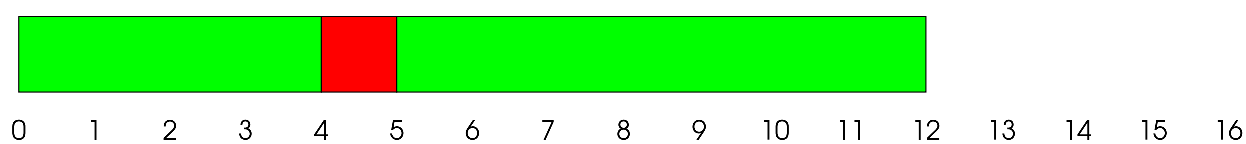
An α-approximation algorithm:

1. The algorithm runs in polynomial time.
2. The algorithm always produces a feasible solution.
3. The value is within a factor α of the optimal value

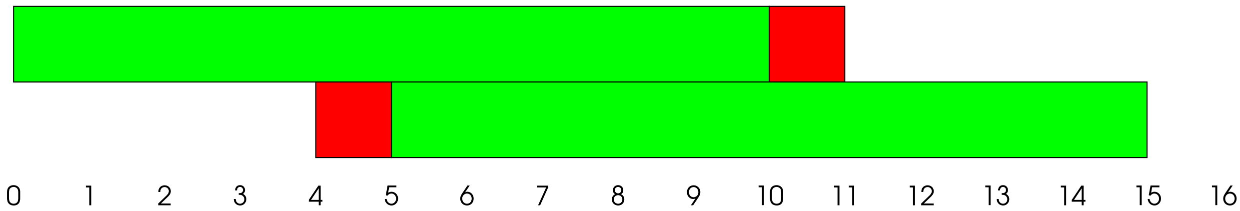
Maximization problem	for all instances: ALG $\geq \alpha \cdot$OPT with $\alpha \leq 1$ 
Minimization problem	for all instances: ALG $\leq \alpha \cdot$OPT with $\alpha \geq 1$ 

jobs j	1	2
released date r_j	0	4
length p_j	10	1

In $1\ r_j \mid \sum_j c_j$, this schedule isn't allowed since haven't been preemptions.



Here are some possible schedule:



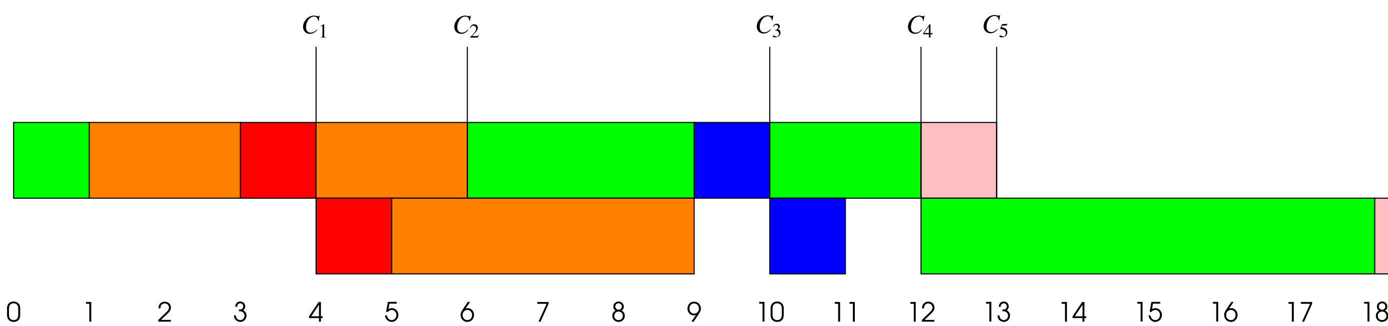
Theorem Problem $1\ r_j\ |\sum_j c_j$ is NP-hard.

Let's see a A 2-approximation algorithm:

Step 1 Apply Shortest Remaining Processing Time (SRPT).
Step 2 Label jobs by completion time in SRPT schedule: $C_1 < \cdots < C_n$. For $j = 1, 2, \ldots, n$: Schedule job j as early as possible after time C_j

Let's see this through an example:

jobs j	1	2	3	4	5
release time r_j	3	1	9	0	12
length p_j	1	4	1	6	1



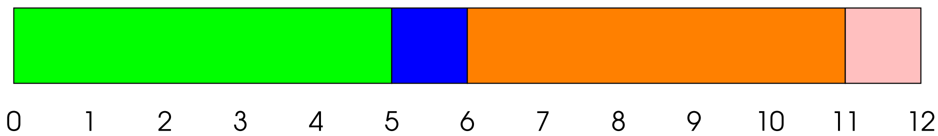
Proof. Proof of approximation ratio 2:
Let C_j be the completion time of job j in SRPT schedule and C'_j is the completion time of job j in final schedule.We can see that:

1. $\sum_j C_j \le \text{OPT}$
2. $p_1 + \cdots + p_j \le C_j$
3. In the final schedule, between time C_j and C'_j there is no idle time
4. In the final schedule, between time C_j and C'_j there are only jobs $k \le j$.
   Therefore
   $$C'_j \le C_j + (p_1 + \cdots + p_j) \le 2C_j$$
   By summing over j, we have
   $$\sum_j C'j \le 2\sum_j C'j \le 2 \cdot \text{OPT}$$

$\square$

jobs j	1	2	3	4
length p_j	5	5	1	1

$$\begin{tikzcd} 1 \arrow[r] \arrow[dr] & 2 \arrow[r] & 3 \\ & 4 & \end{tikzcd}$$



Theorem $1 \mid prec \mid \sum_j C_j$ is NP-hard

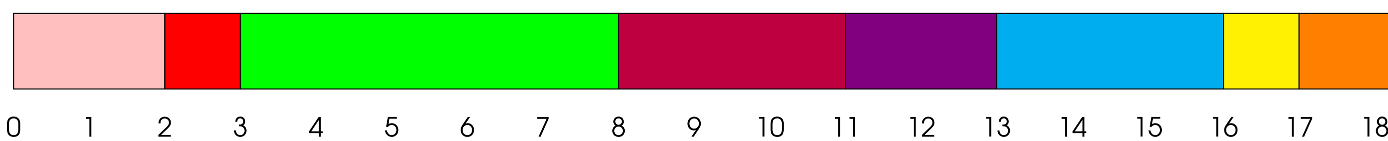
P | pmtn | $C_{\max}$

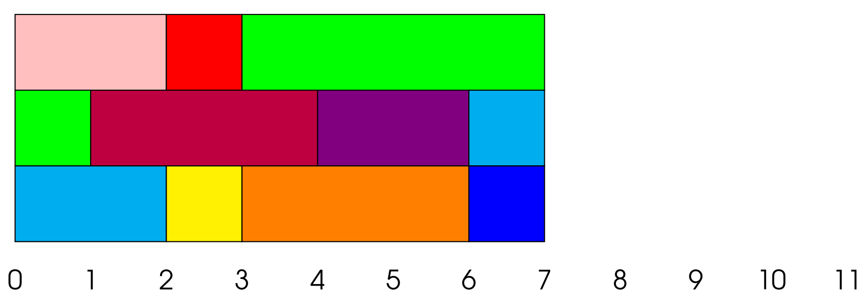
McNaughton's wrap around rule:

1. Calculate the optimal makespan value $C_{\max}^{OPT} = \max(p, \sum_{j=1}^{n} p_j/m)$
2. Construct a single-machine nonpreemptive schedule (assign n jobs to a single machine in an arbitrary order starting with the longest job)
3. Cut this single-machine into m parts of length $C_{\max}^{OPT}$

jobs j	1	2	3	4	5	6	7	8	9
length p_j	2	1	5	3	2	3	1	3	1

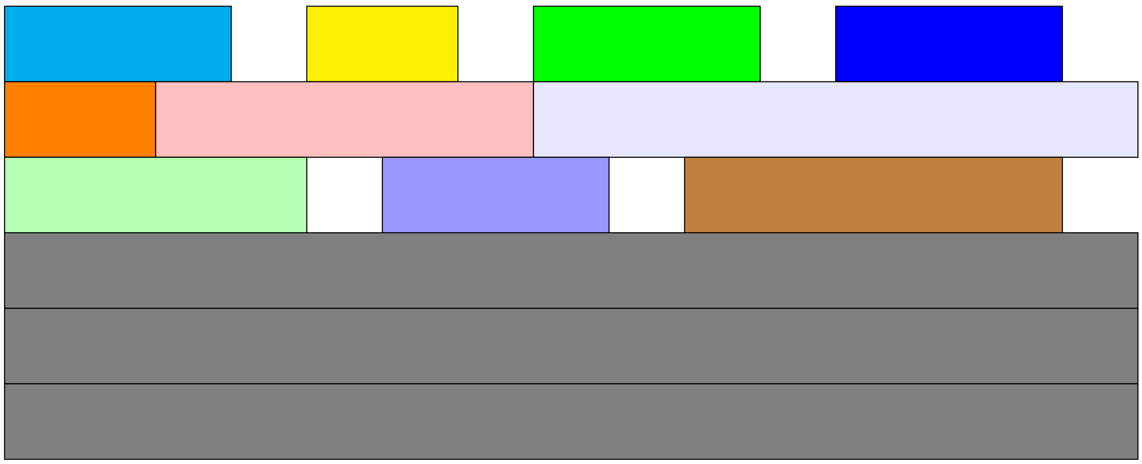
Here $C_{\max}^{OPT} = \max(2/3+1/3+5/3+1+2/3+1+1/3+1+1/3, 5) = \max(7,5) = 7$.

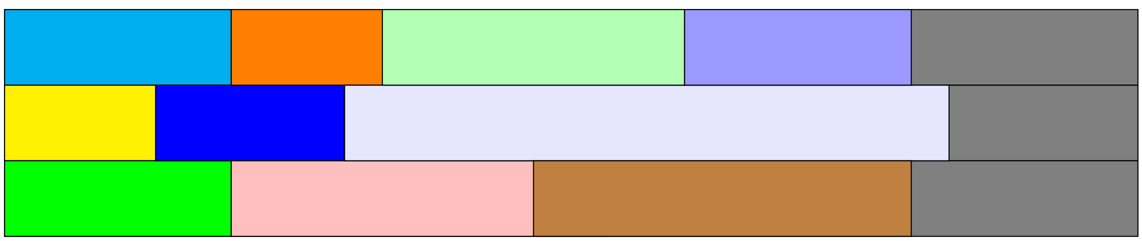




$$P \mid \cdot \mid C_{\max}$$

Scheduling jobs on a identical parallel machines:
1. Start with a list L containing all jobs in some order.
2. Assign the jobs one by one (in the order given by L) to the machines.
3. At any step, choose the machine with the smallest load sofar.





Theorem List scheduling is a $(2-\frac{1}{m})$-approximation algorithm

Proof. Let be L the last job to finish and S_L its starting time.

$C_{\max}$ corresponds to the completion times of the last job to leave the system:

$$C_{\max} = S_L + p_L$$

The List scheduling algorithm implies that before S_L, all machines were busy.Therefore:

$$S_L \times m \leq \sum_{j=1}^{n} p_j - p_L \Longleftrightarrow S_L \leq \frac{p_1 + \cdots + p_n - p_L}{m}$$

The optimal makespan $C_{\max}^{OPT} = (\max(p_1, \ldots, p_n), \sum_{j=1}^{n} p_j/m) \geq \sum_{j=1}^{n} p_j/m$
We can deduce:

$$C_{\max} = S_L + p_L \leq \frac{p_1 + \cdots + p_n}{m} + p_L(1 - \frac{1}{m}) \leq C_{\max}^{OPT} + C_{\max}^{OPT}(1 - \frac{1}{m}) = C_{\max}^{OPT}(2 - \frac{1}{m})$$

□

By using the fact $\max(p_1, \ldots, p_n) \geq p_L$

Theorem In $1 \mid \cdot \mid C_{\max}$ when we apply the list scheduling in the order $p_1 \geq \cdots \geq p_n$, the LPTF (Longest Processing Time first) is a $\frac{4}{3}$-approximation algorithm.

Proof. We want to proove that the length of the LPT schedule is at most 4/3 times the optimal length. Let be L the last job to finish.Notice that we have $p_1 \geq \cdots \geq p_n$ therefore p_L is the minimum processing time possible.

If $p_L \leq C_{\max}^{OPT}/3$, we can reuse the bound from the previous proof:

$$C_{\max} = S_L + p_L \leq \frac{p_1 + \cdots + p_n}{m} + p_L(1 - \frac{1}{m}) \leq C_{\max}^{OPT} + p_L(1 - \frac{1}{m}) \leq C_{\max}^{OPT} + \frac{C_{\max}^{OPT}}{3}(1 - \frac{1}{m})$$

And finally we have:

$$C_{\max} \leq C_{\max}^{OPT}(\frac{4}{3} - \frac{1}{m})$$

Now we have to deal with the case $p_L > C_{\max}^{OPT}/3$.

Since $C_{\max}^{OPT} < 3 \cdot p_L$, $p_L = \min(p_1, \ldots, p_n)$ and $max(p_1, \ldots, pn) \leq C_{\max}^{OPT}$ we have:

$$p_{\max} \leq C_{\max}^{OPT} < 3p_{\min}$$

That means that at most 2 tasks are processed on each machine. It's just a special case in which LPT is optimal.

□

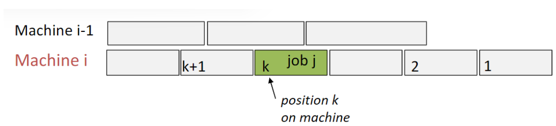
$$R \mid \cdot \mid C_{\max}$$

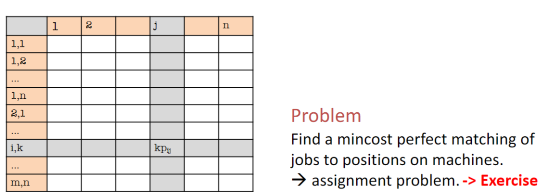
p_{ij}	1	2
$i = 1$	1	2
$i = 2$	2	1

It's clear that put the job 1 on machine 1 and the job 2 in the machine 2 is more optimal than the job 1 on machine 2 and the job 2 in the machine 1.

$R \mid \cdot \mid C_{\max}$ can be reduced to the linear assignment problem.

If job j is scheduled on machine i on position k then it contributes exactly $k \cdot p_{ij}$ to the sum of completion times.





1.1 Tutorial

Exercise 1 Consider the following instance of the scheduling problem $1 \mid \cdot \mid \sum_j C_j \cdot w_j$:

jobs j	1	2	3	4
weight w_j	6	11	9	5
length p_j	3	5	7	4

Give an optimal schedule and its value

Exercise 2 Consider the following instance of the scheduling problem $1 \mid d_j \mid L_{\max}$:

jobs j	1	2	3	4
p_j	5	4	3	6
d_j	3	5	11	12

Give an optimal schedule and its value

Exercise 3 Decision problems Partition and 3-Partition are both NP-complete and are defined as follows:

PARTITION:	An instance is given by positive numbers A and $a_1, a_2, \ldots, a_n$ with $\sum_i a_i = 2A$ Is there an$S \subset \{1,2,...,n\}$ such that $\sum_{i \in P} a_i = A$?
3-PARTITION:	An instance is given by positive numbers B and $b_1, b_2, \ldots, b_{3m}$ with $\sum_i b_i = mB$. Is there a partition of $\{1,2,\ldots,3m\}$ into $S_1, S_2, \ldots, S_m$ such that $\sum_{j \in S_i} b_j = B$? for all $i = 1, \ldots, m$?

(a) Show that the Partition problem can be reduced to the scheduling problem $P_2||C_{\max}$.
(b) Show that the 3-Partition problem can be reduced to the scheduling problem $P||C_{\max}$.

Exercise 4 Consider the scheduling problem $1|r_j|\sum_j Cj$ and the following algorithm (SPT):

When the machine is not processing any job, then start the job that has the smallest processing time pj among the available jobs. (We say that a job is available if it has been released but not started yet). Show by an example that this algorithm does not always lead to an optimal schedule.

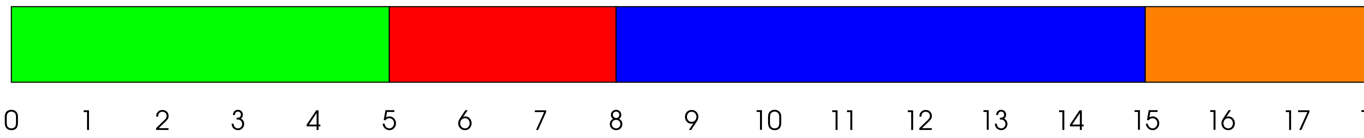
Exercise 5 Consider the scheduling problem $P|r_j, pmtn|C_{\max}$.
Give a polynomial time algorithm which solves the problem by formulating it as a linear program (LP). Assume for simplicity that $0 = r_1 \leq r_2 \leq \cdots \leq r_n$ where n is the number of jobs.

Hint: Use a variable Z for the length of the schedule. The objective then becomes: minimize Z. Take as variables $x_{tj} (t = 1,2,\ldots,n)$ which denote the amount of time spent on job j between time r_t and $rt+1 (t \leq n-1)$ and between rn and $Z (t = n)$. Explain how an optimal LP-solution can be translated into a feasible schedule.

Exercise Show by an example that SRPT is not optimal for $P \mid r_j, pmtn \mid \sum_j c_j$

Solution 1

$$\frac{11}{5} > \frac{6}{3} > \frac{9}{7} > \frac{5}{4}$$



The value is $w_1C_1 + w_2C_2 + w_3C_3 + w_4C_4 = 6 \cdot 8 + 11 \cdot 5 + 9 \cdot 15 + 5 \cdot 19 = 333$

Solution 2 Earliest Due Date: (1,2,3,4)

$L_i = C_i - d_i$

i	1	2	3	4
C_i	5	9	12	18
L_i	2	4	1	6

$L_{\max} = \max(2,4,1,6) = 6$

Solution 3 There is an S with $\sum_{i \in S} a_i = A \iff$ There is a schedule of length $\leq A$ There is a 3-Partition $\iff$ There is a schedule of length $\leq A$

Solution 4

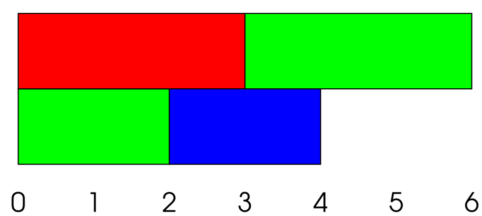
j	1	2
p_j	1	10
r_j	1	0

If we follow the algorithm, we have the order (2,1) $C_2 + C_1 = 10 + 11 = 21$ But if do (1,2), we have$C_1 + C_2 = (1+1) + (2+10) = 14$

Solution 5

Solution

j	1	2	3
p_j	5	3	2
r_j	0	0	2



Completion time for SPT: $3 + (3+6) + 4 = 16$
But this scheduling gives :$5 + 3 + (2+3) = 13$