Flow Shop

In a ***flow shop*** problem, there are m machines that should process n jobs. All jobs have the same processing order through the machines. The order of the jobs on each machine can be different.

In Sections 1-3 we consider the problem of minimising the makespan.

- If there are m=2 machines, then the problem can be solved in $O(n \log n)$ time by Johnson's algorithm.
- If there are m=3 machines or more, then the problem is NP-hard. We discuss the properties of an optimal schedule for the general case with m machines and describe two approximation algorithms.

The problem of minimising the sum of completion times is discussed briefly in Section 4.

1. Example of a flow shop problem

Consider the following instance of problem $F3||C_{\max}$.

j	$p_{1,j}$	$p_{2,j}$	$p_{3,j}$
1	4	2	1
2	3	6	2
3	7	2	3
4	1	5	8

If every machine processes the jobs in the order (1,2,3,4), then the completion times of three operations of job 1 can be calculated as follows:

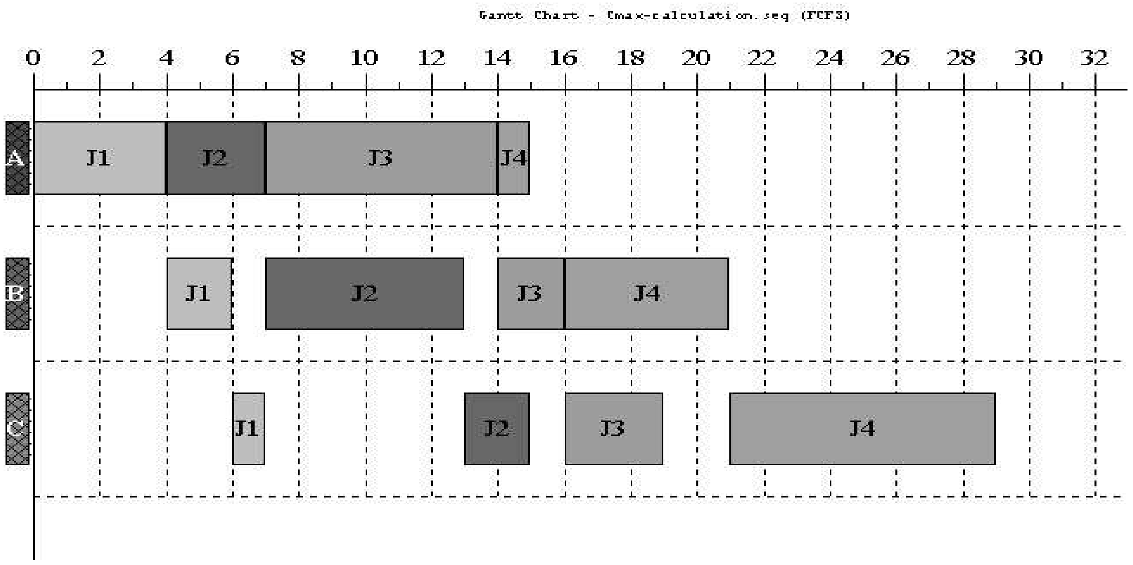
j	$C_{1,j}$	$C_{2,j}$	$C_{3,j}$
1	4	4+2=6	6+1=7
2			
3			
4			

We can continue calculations for jobs 2, 3 and 4:

j	$C_{1,j}$	$C_{2,j}$	$C_{3,j}$
1	4	6	7
2	4+3=7	max{7,6}+6=13	
3			
4			

The makespan of the schedule corresponds to the completion time of the last operation and it is equal to 29.

The Gantt chart of the corresponding schedule is given on the next page.



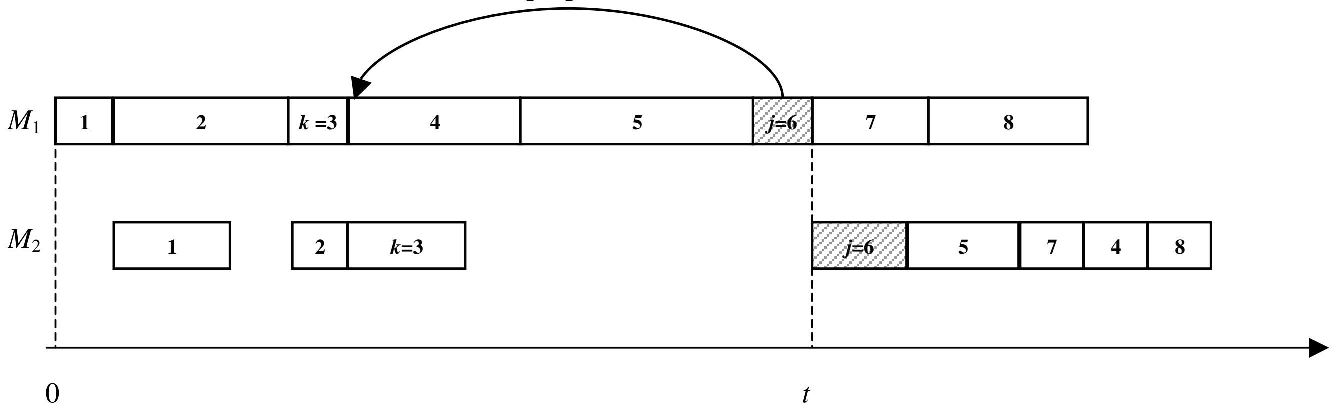
2. Flow shop problem with m=2 machines

We first demonstrate that

> For problem $F2||C_{\max}$ with m**=2** machines there exists an optimal schedule with the same job sequence on both machines.

Suppose there exists an optimal schedule S in which the processing order on the two machines is different.

Let the first k jobs be processed in the same order on both machines. Besides, let job j be processed on machine M_2 in position k+1 and on machine M_1 in position k+1+q. Then we may have a situation as shown in the following figure.



If on machine M_1 we insert job j in-between jobs k and k+1, we get another feasible schedule without postponing any job on machine M_2.

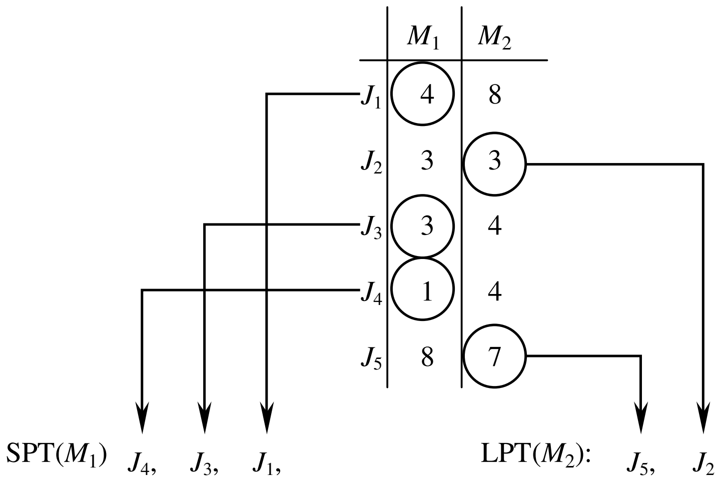
Applying repeatedly this transformation we can finally obtain a schedule with the same sequence of jobs on both machines and with the same or smaller value of the makespan.

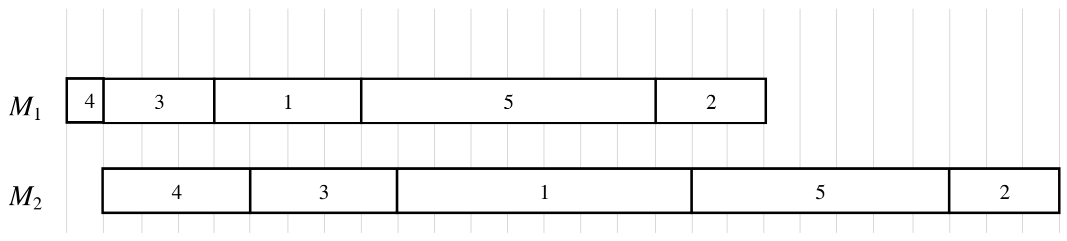
The optimal schedule for the problem $F2||C_{\max}$ can be found in $O(n \log n)$ by the famous algorithm due to Johnson. It was first presented in 1954 and is usually considered as the first result of scheduling theory.

Johnson's algorithm for $F2||C_{\max}$

1. Partition the jobs into two sets with set N_1 containing the jobs with $p_{1j} < p_{2j}$ and set N_2 containing the jobs with $p_{1j} \geq p_{2j}$.
2. The jobs from set N_1 go first, and they go in increasing order of p_{1j} (SPT on M_1).
3. The jobs from set N_2 follow in decreasing order of p_{2j} (LPT on M_2).

Example:





3. Permutation schedules vs. non-permutation schedules

In the ***permutation flow shop*** each machine processes the jobs in the same order. As we have shown in Section 1, for problem $F2||C_{\max}$ there exists an optimal ***permutation*** schedule. In general, the permutation schedule can be non-optimal.

<u>*Example:*</u>

For the following problem with $\boldsymbol{m}$=4 machines and $\boldsymbol{n}$=2 jobs, construct

a) two permutation schedules S_1 and S_2;

b) a non-permutation schedule S_3.

Verify that $C_{\max}(S_3)$ is less than $C_{\max}(S_1)$ and $C_{\max}(S_2)$.

	M_1	M_2	M_3	M_4
J_1	4	1	1	4
J_2	1	4	4	1

	M_1																					
	M_2																					
$\boldsymbol{S_1}$**:**	M_3																					
	M_4																					
	M_1																					
	M_2																					
$\boldsymbol{S_2}$**:**	M_3																					
	M_4																					
	M_1																					
	M_2																					
$\boldsymbol{S_2}$**:**	M_3																					
	M_4																					

Two permutation schedules: $C_{\max}(S_1) =$

$C_{\max}(S_2)=$

Non-permutation schedule: $C_{\max}(S_3)=$

3. Approximation algorithms for $F||C_{\max}$

While the flow shop problem $F2||C_{\max}$ with m=2 machines is solvable in polynomial time by Johnson's algorithm, the problem $F||C_{\max}$ with m≥3 machines is NP-hard.

The two approximation algorithms presented below for solving the flow shop problem with m machines are based on Johnson's algorithm for the two-machine problem $F2||C_{\max}$. Each of them has performance guarantee $\rho = \lceil m/2 \rceil$.

Decomposition Algorithm *FSDecomp*

1. If m is odd, add new machine m+1 and set $p_{m+1,j}=0$, $j=1,\ldots n$.
2. Divide the set of the machines into $m/2$ pairs. For each pair of machines $2k$-1, $2k$, $k=1,\ldots,m/2$, find the optimal schedule S_k using Johnson's algorithm.
3. Using the schedules S_k, $k=1, \ldots,m/2$, construct a schedule S for the original problem.

Aggregation Algorithm *FSAggr*

1. If m is odd, add new machine m+1 and set $p_{m+1,j}=0$, $j=1,\ldots n$.
2. Consider the problem with two artificial machines A and B and the processing times equal to $a_j = \sum_{i=1}^{m/2} p_{ij}$, $b_j = \sum_{i=m/2+1}^{m} p_{ij}$. Find the optimal two-machine schedule s_{AB} using Johnson's algorithm.
4. Using the schedule s_{AB}, construct a permutation schedule s for the original problem.

Exercise 1:
Consider flow shop problem with n=5 jobs and m=4 machines.
Determine the job sequence according to algorithm *FSDecomp*

	M_1	M_2	M_3	M_4
J_1	5	4	4	3
J_2	5	4	4	6
J_3	3	2	3	3
J_4	6	4	4	2
J_5	3	4	1	5

Exercise 2:
Now we solve the same problem by algorithm *FSAggr*.

	M_1	M_2	M_3	M_4
J_1	5	4	4	3
J_2	5	4	4	6
J_3	3	2	3	3
J_4	6	4	4	2
J_5	3	4	1	5

4. Approximation algorithm for $F||\Sigma C_j$

Problem $F2||\Sigma C_j$ is NP-hard even for the case of m=2 machines.

The following algorithm to minimise ΣC_j is due to Gonzalez & Sahni.
It has a ratio guarantee $\rho=m$.

Algorithm by Gonzalez & Sahni (1978) (SPT-rule)

1. Number the jobs such that $p_1 \le p_2 \le \ldots \le p_n$, where $p_j = \sum_{i=1}^{m} p_{ij}$.
2. Construct a permutation schedule such that all machines process the jobs according to the permutation $(1,2,\ldots,n)$.