Tutorial 3 : Production planning

Operations research, 3rd semester.

2024

Exercice 1 — *Flow Shop Problem*

Five jobs must be performed, on a machine M_1 and on a machine M_2, in any order. When the task of a job j on a machine is done, the other machine can start to work on the job j if and only if it is not currently working on another job. In that case, we must wait for the machine to be available before using it for j.

1. Give the optimal value and an optimal solution if the duration of each job on each machine is given on the following table.

Duration $t(j,M)$	j_1	j_2	j_3	j_4	j_5
M_1	2	4	1	8	2
M_2	1	3	2	10	1

▶ **Correction**

We are in the case where both machines have no priority order. In this case, we need to calculate M and T to decide. Here, $M > T$, with $M = t(j_4, M_1) + t(j_4, M_2)$. Thus, we have the following solution.

M_1	4	4	4	4	4	4	4	4	1	1	2	2	2	2	3	5	5	
M_2	1	2	2	2	3	3	5		4	4	4	4	4	4	4	4	4	4

2. Same question with the following durations.

Duration $t(j,M)$	j_1	j_2	j_3	j_4	j_5
M_1	3	4	2	5	3
M_2	4	3	2	1	5

▶ **Correction**

We are now in the case where $M < T$. We have $T = 17 = T_1$ and $T_2 = 15$. We will add 2 dummy jobs to make $T_1 = T_2$.

Duration $t(j,M)$	j_1	j_2	j_3	j_4	j_5	j_6	j_7
M_1	3	4	2	5	3	0	0
M_2	4	3	2	1	5	1	1

We then group the tasks into 3 sets J_1, J_2, J_3 using the algorithm discussed in class. We then obtain

$$J_1 = \{j_1, j_2, j_6, j_7\}, J_2 = \{j_3, j_4\}, J_3 = \{j_5\}$$

We have the following table and find an optimal solution with a value of 17. As $t(J_1, M_1) > t(J_2, M_2)$ and $t(J_1, M_2) > t(J_3, M_1)$, we have

Duration $t(j, M)$	J_1	J_2	J_3
M_1	7	7	3
M_2	9	3	5

M_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_2	J_2	J_2	J_2	J_2	J_2	J_2	J_3	J_3	J_3
M_2	J_2	J_2	J_2	J_3	J_3	J_3	J_3	J_3	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1

arg2

- $J_1 = \{j_3, j_4\}, J_2 = \{j_5\}, J_3 = \{j_1, j_2, j_6, j_7\}$
- $J_1 = \{j_5\}, J_2 = \{j_1, j_2, j_6, j_7\}, J_3 = \{j_3, j_4\}$

We then go back to the original jobs by replacing each set with the jobs in it.

M_1	1	1	1	2	2	2	2	3	3	4	4	4	4	4	5	5	5
M_2	3	3	4	5	5	5	5	5	1	1	1	1	2	2	2	6	7

Exercice 2 — *Flow Shop Problem (bis)*

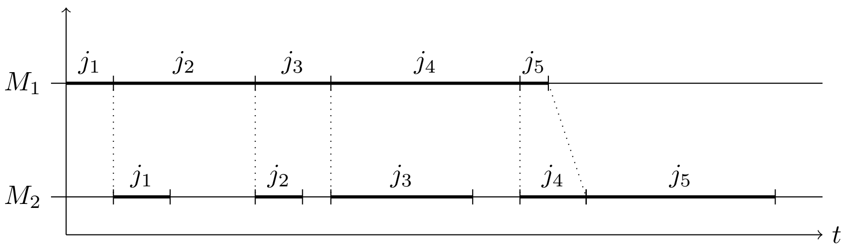
Five jobs must be performed, firstly on a machine M_1 and then on a machine M_2, in that order. When the task of a job j on M_1 is done, the machine M_2 can start to work on the job j if and only if it is not currently working on another job. In that case, we must wait for M_2 to be available before using it for j. The duration of each job on each machine is given on the following table.

Duration $t(j, M)$	j_1	j_2	j_3	j_4	j_5
M_1	50	150	80	200	30
M_2	60	50	150	70	200

1. What is the total duration if we perform the jobs j_1, j_2, j_3, j_4, j_5 in that order.
2. Apply the Johnson algorithm in order to find an order that minimize the total duration of the execution.
3. What is the complexity of that algorithm ?
4. We assume the following fact as proved : a solution is optimal if for every couple of jobs i preceding j, $\min(t(i, M_1), t(j, M_2)) \leq \min(t(i, M_2), t(j, M_1))$. Show that a solution returned by the Johnson algorithm is optimal.

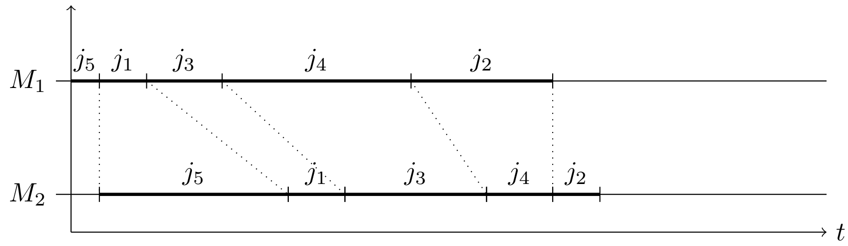
▶ Correction

1. Here is the associated GANTT diagram :



The total duration we get is 750.

2. We obtain $A = [j_1, j_3, j_5]$ and $B = [j_2, j_4]$. We sort A according to M_1 and B according to M_2 in descending order : $S_A = [j_5, j_1, j_3]$ and $S_B = [j_4, j_2]$. This gives the order $S = [j_5, j_1, j_3, j_4, j_2]$. We can redo a GANTT to ensure that it works. The duration is 560.



3. The complexity of Johnson's algorithm is $O(n \log n)$ where n is the number of tasks. Indeed, the algorithm begins with a loop that has n iterations, each performing a comparison and an addition to a list in $O(1)$, followed by two sorts in $O(n \log n)$, and finally a concatenation in $O(1)$ or $O(n)$ depending on the chosen list structure (linked list or array).
4. We want to verify that the solution returned by the algorithm satisfies this property.
   There are 3 cases to check :
   — $i \in A$ and $j \in A$, and i is before j in S,

$$\begin{aligned} t_1(i) &\leq t_2(i) \\ t_1(i) &\leq t_1(j) \\ \Rightarrow t_1(i) &\leq \min(t_2(i), t_1(j)) \\ \min(t_1(i), t_2(j)) &\leq \min(t_2(i), t_1(j)) \end{aligned}$$

   — $i \in A$ and $j \in B$

$$\begin{aligned} t_1(i) &\leq t_2(i) \\ t_2(j) &\leq t_1(j) \\ \Rightarrow \min(t_1(i), t_2(j)) &\leq t_2(i) \\ \text{and } \min(t_1(i), t_2(j)) &\leq t_1(j) \\ \Rightarrow \min(t_1(i), t_2(j)) &\leq \min(t_2(i), t_1(j)) \end{aligned}$$

   — $i \in B$ and $j \in B$, and i is before j in S, :

$$\begin{aligned} t_2(i) &\leq t_1(i) \\ t_2(j) &\leq t_2(i) \\ \Rightarrow t_2(j) &\leq \min(t_2(i), t_1(j)) \\ \min(t_1(i), t_2(j)) &\leq \min(t_2(i), t_1(j)) \end{aligned}$$

   The last possible case ($i \in B$ and $j \in A$) would be in contradiction with the fact that i is before j in S.

Exercice 3 — *Flow Shop Problem with 3 machines*

Same subject as the previous exercice but with 3 machines.

$t(j, M)$	j_1	j_2	j_3	j_4	j_5	j_6
M_1	60	40	80	70	100	50
M_2	40	20	10	30	20	30
M_3	40	60	70	100	50	80

1. We consider a fictive problem with only two machines M'_1 and M'_2 where, for each job j, $t(j, M'_1) = t(j, M_1) + t(j, M_2)$ and $t(j, M'_2) = t(j, M_2) + t(j, M_3)$. Apply the Johnson algorithm in order to find an optimal solution on that fictive problem.
2. What is the total duration of the execution of the fictive problem ?

3. What is the total duration of that same execution of the real problem ?
4. Reuse the algorithm on the following table

$t(j, M)$	j_1	j_2	j_3
M_1	10	20	40
M_2	70	80	20
M_3	20	00	35

5. Compare this solution with the following ordering $j_3 j_1 j_2$. Is this algorithm always optimal ?
6. The algorithm is optimal for the first example but not the second. What could explain this ?

► **Correction**

1. We have the following table

Duration $t(j, M)$	j_1	j_2	j_3	j_4	j_5	j_6
M'_1	100	60	90	100	120	80
M'_2	80	80	80	130	70	110

We get the order $S = [j_2, j_6, j_4, j_1, j_3, j_5]$ or $S = [j_2, j_6, j_4, j_3, j_1, j_5]$

2. We find 620, below is a discrete version of the Gantt chart (a number corresponds to a slot of 10 time units, an X is a moment when M_2 is not operating.)

```
M1      2222226666666644444444441111111111333333333555555555555
M2      XXXXXX2222222266666666666444444444444411111111333333333X5555555
       |         |         |         |         |         |         |
Temps  0        100       200       300       400       500       600
```

3. We find 470. Note : it is optimal, but nothing intuitive proves that. Below is the associated Gantt chart.

```
M1      2222666664444444111111333333335555555555
M2      XXXX22XXX666XXXX444XXX1111XXXX3XXXXXXXXX55
M3      XXXXXX22222266666666444444444411113333333X55555
       |         |         |         |         |
Temps  0        100       200       300       400
```

4. We get thge following table.

Duration $t(j, M)$	j_1	j_2	j_3
M'_1	80	100	60
M'_2	90	80	55

We get the following order : $S = [j_1, j_2, j_3]$.

5. The previous order gives a total duration of 215, while j_3, j_1, j_2 gives 210.
6. The difference between the two instances is that, in the second, the machine M_2 is bounded by M_1 and M_3. It is said to be dominated. In this case, it has been proven that Johnson's algorithm gives the optimal result.

Exercice 4 — *Project planning under resources constraints - heuristic.*

We consider the following project planning problem where the number of employees is 5.

task	duration	previous tasks	nb employees
A	6	-	3
B	3	-	2
C	6	-	1
D	2	B	1
E	4	B	3
F	3	D A	3
G	1	F E C	2

The PERT/Metra potential methods does not take into account any capacity constraint as the number of employees. The following algorithm, called the serial method or list algorithm, solve the problem. However it may not return an optimal solution : this is called a heuristic.

a) Define a priority order for the tasks satisfying the precedence constraint.
b) For each task t, in that order
   A) Consider the minimum time τ where the task t may be done (precedence and ressource constraint).
   B) Define the starting time of the task t as τ.

We are going to use this heuristic for our example.

1. By using the metra potential method, find an optimal ordering that ignore the resource constraint. Is this ordering satisfying that constraint ? Compute the late start of each task.
2. Apply the serial method. The priority of each task is its late start.
3. What is the GANTT diagram of that planning ?

▶ **Correction**

1. Optimal scheduling by earliest times :
   A : 0 - 6
   B : 0 - 3
   C : 0 - 6
   D : 3 - 5
   E : 3 - 7
   F : 6 - 9
   G : 9 - 10

   There are therefore 6 employees working with A, B, and C at the start, which is excluded.
   Optimal scheduling by latest times :
   A : 0 - 6
   B : 1 - 4
   C : 3 - 9
   D : 4 - 6
   E : 5 - 9
   F : 6 - 9
   G : 9 - 10

   There are 6 employees working with A, B, and C at time 3, which is excluded.
2. We get the following order : A ; B ; C ; D ; E ; F ; G.
   We have the following schedules
   A : 0 - 6
   B : 0 - 3
   C : 3 - 9 nb employees : cannot be done before B ends
   D : 3 - 5 nb employees and B should end before D starts.
   E : 6 - 10 nb employees : cannot be done before A ends
   F : 10 - 13 nb employees : cannot be done before E ends

   G : 13 - 14 (F is before G)