TD4 : Ordonnancement et Planification de la production par optimisation

- Planification du Flow Shop (modèle donné)
- Planification du Job Shop (à modéliser)

A. Planification du Flow Shop

Considérons un atelier de production avec m machines où chaque machine k doit traiter n tâches. Chaque tâche doit être traitée dans un ordre fixe $\{M_1, M_2, \dots, M_m\}$, c'est-à-dire que toutes les tâches doivent passer successivement à travers chaque machine dans la même séquence. L'objectif est de minimiser le makespan tout en tenant compte des temps d'inactivité des tâches et des intervalles d'inactivité des machines. Pour cela, on doit choisir la séquence des tâches, par exemple $S = \{T_9, T_2, T_{10}, \dots, \}$ qui minimise le makespan.

Le problème d'optimisation est le suivant :

- x_{ik} : variables binaires de position dans la séquence (somme de lignes =1 / somme colonnes =1) ;
- p_{ki} : le temps de traitement de la tâche i dans la machine k
- I_{kj} : intervalle d'inactivité sur la machine k avant le début de la tâche à la position j ;
- H_{kj} : temps d'inactivité de la tâche dans la position j après avoir fini dans la machine k ;

A partir de l'achèvement de la tâche à la position j sur la machine k, nous pouvons mesurer le temps jusqu'au début de la tâche à la position $j+1$ sur la machine $k+1$ de deux manières.

a. Nous pouvons ajouter l'intervalle d'inactivité sur la machine k avant le début de la tâche à la position $j+1$, le temps de traitement sur la machine k de la tâche à la position de séquence $j+1$, et le temps d'inactivité pour la tâche à la position de séquence $j+1$ après la fin sur la machine k. Cela correspond à la somme de trois termes :

$$Val = I_{k,j+1} + \sum_{i=1}^{n} p_{ki} x_{i,j+1} + H_{k,j+1}$$

b. Nous pouvons ajouter le temps d'inactivité pour la tâche à la position de séquence j après la fin sur la machine k, le temps de traitement sur la machine $k+1$ de la tâche à la position de séquence j, et l'intervalle d'inactivité sur la machine $k+1$ avant le début de la tâche à la position de séquence $j+1$. Cela correspond également à la somme de trois termes :

$$Val = H_{k,j} + \sum_{i=1}^{n} p_{k+1,i} x_{ij} + I_{k+1,j+1}$$

Donc,

$$I_{k,j+1} + \sum_{i=1}^{n} p_{ki} x_{i,j+1} + H_{k,j+1} - H_{k,j} - \sum_{i=1}^{n} p_{k+1,i} x_{ij} - I_{k+1,j+1} = 0$$

pour toutes les positions $j = 1, \dots, n-1$ et toutes les machines $k = 1, \dots, m-1$.
Pour la première tâche

$$I_{k1} + \sum_{i=1}^{n} p_{ki} x_{i1} + H_{k1} - I_{k+1,1} = 0$$

pour toutes les machines $\boldsymbol{k = 1, \dots, m-1}$.
Pour minimiser le makespan :

$$\min \mathrm{M} = \sum_{i=1}^{n} p_{mi} + \sum_{j=1}^{n} I_{mj}$$

La première somme dans cette expression, représentant le temps total de traitement requis sur la dernière machine, est simplement une constante. Ainsi, pour minimiser le temps d'exécution total, nous devons minimiser la somme des temps d'inactivité sur la dernière machine, $\sum_{j=1}^{n} I_{mj}$.

Considérez les données suivantes (6 tâches, 3 machines, le temps de traitement p_{ki} de la tâche i par la machine/opération k) :

Tâche	**1**	**2**	**3**	**4**	**5**	**6**
Opération 1	75	36	62	8	25	32
Opération 2	43	48	26	10	12	83
Opération 3	67	50	18	37	18	57

Trouver la séquence S qui minimise le critère d'optimisation M. Fournir le script.

B. Planification du Job Shop

Le problème classique de planification du Job Shop diffère du problème de Flow Shop sur un point important : le flux de travail n'est pas unidirectionnel. Les éléments du problème comprennent un ensemble de m machines et une collection de n tâches à planifier. Chaque tâche se compose de plusieurs opérations avec la même structure de précédence linéaire que dans le modèle de flow shop. Bien qu'une tâche puisse avoir un

nombre quelconque d'opérations, la formulation la plus courante du problème de Job Shop spécifie que chaque tâche a exactement m opérations, une sur chaque machine.

Dans le flow shop, la machine k effectue la k-ème opération de n'importe quelle tâche, et il n'est pas nécessaire de faire la distinction entre l'opération et la machine. En revanche, dans le job shop, il est approprié de décrire une opération avec un triplet (i, j, k) pour indiquer que, pour la tâche i, l'opération j nécessite la machine k.

A présent nous utiliseront le modèle disjonctif : chaque tâche j doit suivre un ordre de traitement $(O^{j}_{1}, O^{j}_{2}, \dots, O^{j}_{k})$, et chaque opération (m, j) a une durée de traitement p_{mj}.

Les variables de décision considérées sont :

- le moment où la tâche j commence sur la machine m, noté t_{mj} ;
- une variable binaire qui indique la précédence de la tâche i avant la tâche j sur la machine m, notée x_{mij} ;
- le temps total d'achèvement des opérations, C, qui est lui-même l'objectif de minimisation.

Les contraintes :

- Le temps de début de la tâche j sur la machine m doit être supérieur au temps de début de l'opération précédente de la tâche j plus sa durée de traitement.
- Chaque machine ne traite qu'une tâche à la fois. Pour cela, nous indiquons que si i précède j sur la machine m, le temps de début de la tâche j sur la machine m doit être supérieur ou égal au temps de début de la tâche i plus sa durée de traitement. Sinon, le temps de début de la tâche i plus sa durée de traitement n'est pas bornée : $a + b \leq c + M(1 - x), x \in \{0,1\}$ si $x = 1$ donc $a + b \leq c$ sinon $a + b \leq M$, avec $M = 1000$ par exemple (Big-M).
- Pour chaque paire de tâches i, j, l'un des deux doit précéder l'autre pour chaque machine.
- Le makespan total est supérieur au temps de début de chaque opération plus sa durée de traitement.

Modéliser ce problème et trouver une solution pour un choix arbitraire des données. Fournir le script.

Exemple :

Ordre :

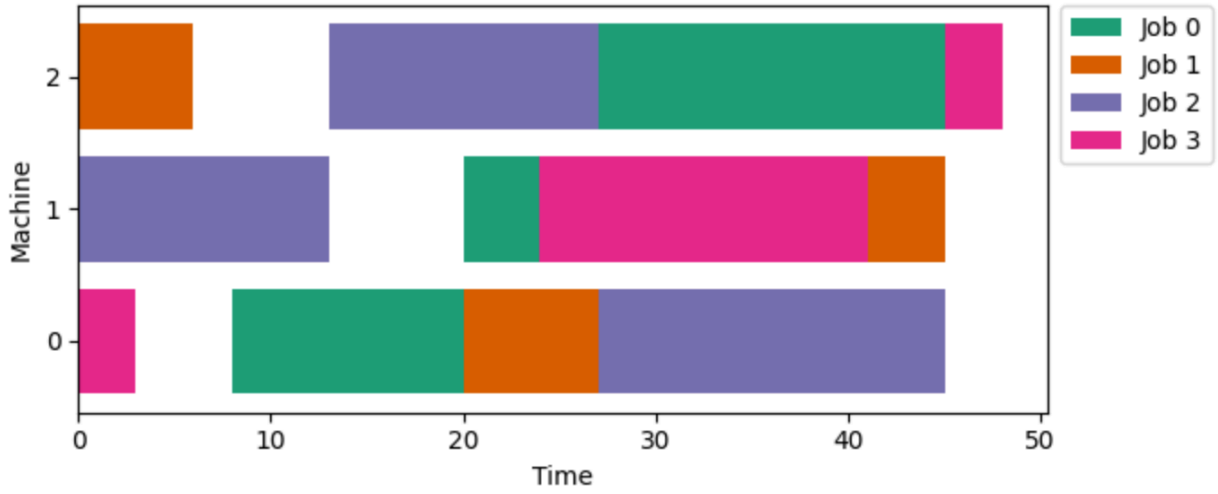
Tâche	**1**	**2**	**3**	**4**
Opération 1	1	3	2	1

Opération 2	2	1	3	2
Opération 3	3	2	1	3

Temps de traitement :

Tâche	**1**	**2**	**3**	**4**
Opération 1	12	7	18	3
Opération 2	4	4	13	17
Opération 3	18	6	14	3

Diagramme de Gantt :



Fonction objective : 48 (mais plusieurs ordonnancements différents donc t_{mj} et x_{mij} peuvent nous donner le même résultat optimal, par exemple essayez de changer le solveur : cplex → gurobi)

`card(M)` → $|M|$

Pour faire : $k \in S \setminus \{j\}$ → `k in S diff{j}` ou `k in S: j!= k`

Pour définir une matrice :

```
param p :=
1 1 12
1 2 7
1 3 18
1 4 3
2 1 4
2 2 4
2 3 13
2 4 17
3 1 18
3 2 6
3 3 14
3 4 3
;
```

ou

```
param p : 1 2 3 4 :=
1 12 7 18 3
2 4 4 13 17
3 18 6 14 3
;
```