Ordonnancement sur machines parallèles

SIO - Laboratoire 2

Nicolas Crausaz & Maxime Scharwath

17.01.2023

Table des matières

Modélisation mathématique

Contexte

L'objectif de ce travail est d'effectuer la modélisation d'un problème d'ordonnancement, consistant à trouver un plan d'ordonnancement permettant de répartir n tâches, devant toutes être réalisée, en disposant de m machines différentes (travail en parrallèle), cela en minimisant le retard moyen de l'éxécution des tâches.

Nous connaissons les **constantes** suivantes:

Pour chaque tâche $i = 1, \ldots, n$:

- Sa **date de disponibilité** (date de début au plus tôt, release date): r_i
- Sa **date d'échéance** (date de fin au plus tard, due date): d_i
- Son **temps d'exécution** (durée de réalisation, processing time): p_i

On supposera, sans perte de généralité, que la plus petite date de disponibilité est égale à 0 et que les données sont cohérentes et vérifient, en particulier, $r_i \geq 0$ et $p_i \geq 0$ pour chaque tâche $i = 1, .., n$.

Définition des variables de décision

Pour trouver un ordonnancement de n tâches sur m machines, il va nous falloir répartir ces tâches de manière à ce qu'une tâche $i = 1, \ldots, n$ soit traitée sur une unique machine $k = 1, \ldots, m$. Pour cela il nous faut donc définir les variables de décision suivantes:

Une variable binaire u_{ik} indiquant si une tâche i est executée sur la machine k.

$$u_{ik} = \begin{cases} 1 & \text{si la tâche i est executée sur la machine k} \\ 0 & \text{sinon} \end{cases}$$

ainsi qu'une seconde variable ayant comme valeur la date début d'un tâche i s'exécutant sur une machine k:

$$x_{ik} = \begin{cases} \text{Date de début de la tâche i sur la machine k} & \text{si i est executé sur k} \\ 0 & \text{sinon} \end{cases}$$

Définition des variables auxiliaires

Afin de pouvoir connaître le retard de chaque tâche sur sa machine respective, nous définissons la variable T_i, correspondant au retard (tardiness) de la tâche $i = 1, \ldots, n$ lors de son exécution:

$$T_i = \sum_{k=1}^{m} (x_{ik}) + p_i - d_i$$

On introduit une variable auxiliaire binaire y_{ij}, pour chaque paire $\{i,j\}$, $i,j = 1,\ldots,n$ de tâches différentes s'exécutant sur une même machine et dont l'interprétation est:

$$y_{ij} = \begin{cases} 1 & \text{si la tâche } i \text{ est executé avant la tâche } j \text{ sur la même machine} \\ 0 & \text{si } i \text{ et } j \text{ ne sont pas exécutés sur la même machine} \\ 0 & \text{sinon} \end{cases} \quad i = 1,\ldots,n, j = 1,\ldots,m$$

qui se linéarise de la manière suivante:

$$y_{ij} = \begin{cases} x_{ik} + p_i - x_{jk} \leq M \cdot (1 - y_{ij}) + M \cdot (1 - u_{ik}) + M \cdot (1 - u_{jk}) \\ x_{ik} + p_i - x_{jk} \leq M y_{ij} + M \cdot (1 - u_{ik}) + M \cdot (1 - u_{jk}) \end{cases} i = 1,\ldots,n, j = 1,\ldots,m$$

que nous pouvons simplifier en:

$$y_{ij} = \begin{cases} x_{ik} + p_i - x_{jk} \leq M \cdot (3 - y_{ij} - u_{ik} - u_{jk}) \\ x_{ik} + p_i - x_{jk} \leq M \cdot (2 + y_{ij} - u_{ik} - u_{jk}) \end{cases} i = 1,\ldots,n, j = 1,\ldots,m$$

dont la constante M devant avoir une valeur suffisamment grande, nous la définissons à:

$$M = \max_{i=1,\ldots,n}(r_i) + \sum_{i=1}^{n} p_i$$

correspondant à la date de disponibilité de la tâche s'exécutant le plus tard additionné à la somme du temps d'exécution de toutes les tâches. Ceci correspond à une valeur plus grande que le retard maximal possible.

Définition de la fonction objectif

Nous recherchons un ordonnancement répartissant n tâches sur m machines différentes, qui minimise le **retard moyen** de l'exécution des tâches.

$$\text{Minimiser} \qquad z = \frac{1}{n}\sum_{i=1}^{n} T_i$$

Avec, pour rappel:

$$T_i = \sum_{k=1}^{m}(x_{ik}) + p_i - d_i, i = 1, \ldots, n$$

Définition des contraintes

Nous allons établir une série de contraintes pour faire respecter la cohérence et les particularités du problème d'ordonnancement:

(1) Une tâche n'est executé qu'une seule fois et sur une unique machine, se traduisant:

$$\sum_{k=1}^{m} u_{ik} = 1, \qquad i = 1, \ldots, n$$

(2) Le retard d'une tâche i (Ti) s'exécutant sur une machine k est postérieur (plus grande) à sa durée d'exécution moins son écheance, ceci se traduit en:

$$T_i \geq \sum_{k=1}^{m}(x_{ik}) + p_i - d_i, \qquad i = 1, \ldots, n$$

(3) L'exécution de chaque tâche ne peut commencer avant sa date de disponibilité:

$$x_{ik} \geq r_i \cdot u_{ik} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, m$$

(4) et (5) Pour chaque paire {i, j} de tâches, soit la tâche i termine son exécution avant que la tâche j ne débute soit c'est l'inverse:

$$x_{ik} + p_i - x_{jk} \leq M \cdot (1 - y_{ij}) + M \cdot (1 - u_{ik}) + M \cdot (1 - u_{jk})$$

$$x_{ik} + p_i - x_{jk} \leq M \cdot y_{ij} + M \cdot (1 - u_{ik}) + M \cdot (1 - u_{jk})$$

$$i = 1, \ldots, n \qquad j = 1, \ldots, n \qquad k = 1, \ldots, m$$

(6) et (7) Contraintes de non négativité:

$$x_{ik}, T_i \geq 0$$

Le PL résultant sera:

$$\text{Minimiser} \qquad z = \frac{1}{n}\sum_{i=1}^{n} T_i$$

s.c

$$
\begin{array}{lcll}
\sum_{k=1}^{m} u_{ik} & = & 1 & (1) \\
T_i & \geq & \sum_{k=1}^{m} (x_{ik}) + p_i - d_i & (2) \\
x_{ik} & \geq & r_i \cdot u_{ik} & (3) \\
x_{ik} + p_i - x_{jk} & \leq & M \cdot (3 - y_{ij} - u_{ik} - u_{jk}) & (4) \\
x_{ik} + p_i - x_{jk} & \leq & M \cdot (2 + y_{ij} - u_{ik} - u_{jk}) & (5) \\
x_{ik}, T_i & \geq & 0 & (6)(7)
\end{array}
$$

$$i = 1, \ldots, n \qquad j = 1, \ldots, n \qquad k = 1, \ldots, m$$