Ordonnancement sur machines parallèles

SIO - Laboratoire 2

Nicolas Crausaz & Maxime Scharwath

10.01.2023

Table des matières

Modélisation mathématique

Le contexte 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 $i = 1, ..., n$:

- 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, $r_i \ >= \ 0$ et $p_i \ >= \ 0$ pour chaque tâche $i = 1, .., n$.

Définition des variables de décision

Nous définissons la variable de décision suivante:

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

Nous définissons la variable binaire:

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

Définition des variables auxiliaires

On defini le retard (tardiness) Ti de la tâche i sur la machine k par - $T_i = \max_{i=1,\ldots,n} (0, \sum_{k=1}^{m} x_{ik} + p_i - d_i)$

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

$$y_{ij} = \begin{cases} 1 & \text{si la tâche est executé avant la tâche j sur la même machine} \\ 0 & \text{sinon} \\ 0 & \text{sinon} \end{cases} \qquad \text{i = 1,...,n, j = 1,...n}$$

Cette contraintes se linéaire de la manière suivante:

Il faut définir la constante M = max ri + somme de tous les pi, ayant une valeur suffisamment grande. (peut etre mettre ça plus haut)

linéarisation: xik + pi - xjk <= M * (1 - yij) + M * (1 - Uik) + M * (1 - Ujk) xik + pi - xjk <= M * (1 - yij) + M * (Uik) + M * (1 - Ujk) xi + pi - xjk <= M * (1 - yij) + M * (1 - Uik) + M * (1 -Ujk) xik + pi - xjk <= M * (1 - yij) + M * Uik + M * Ujk xik + pi - xjk <= M * yij + M * (1 - Uik) + M * (1 - Ujk) xik + pi - xjk <= M * yij + M * (Uik) + M * (1 - Ujk) xik + pi - xjk <= M * yij + M * (1 - Uik) + M * (Ujk) xik + pi - xjk <= M * yij + M * (Uik) + M * (Ujk)

$$y_{ij} = \begin{cases} x_{ik} + p_i - x_{jk} <= M * (1 - y_{ij}) + M * (1 - U_{ik}) + M * (1 - U_{jk}) \\ x_{ik} + p_i - x_{jk} <= M * (1 - y_{ij}) + M * (U_{ik}) + M * (1 - U_{jk}) \\ x_{ik} + p_i - x_{jk} <= M * (1 - y_{ij}) + M * (1 - U_{ik}) + M * (1 - U_{jk}) \\ x_{ik} + p_i - x_{jk} <= M * (1 - y_{ij}) + M * U_{ik} + M * U_{jk} \\ x_{ik} + p_i - x_{jk} <= M * y_{ij} + M * (1 - U_{ik}) + M * (1 - U_{jk}) \\ x_{ik} + p_i - x_{jk} <= M * y_{ij} + M * (Uik) + M * (1 - U_{jk}) \\ x_{ik} + p_i - x_{jk} <= M * y_{ij} + M * (1 - U_{ik}) + M * (U_{jk}) \\ x_{ik} + p_i - x_{jk} <= M * y_{ij} + M * (U_{ik}) + M * (U_{jk}) \end{cases} \quad \text{i,j = 1,...,n, k = 1,...m}$$

TODO

Définition de la fonction objectif

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

Définition des contraintes

- Une tâche n'est executé qu'une seule fois et sur une unique machine

sum Uik = 1, i 1 à n et l 1 à m

- L'exécution de chaque tâche ne peut commencer avant sa date de disponibilité

Xik >= ri * Uik, pour k = 1 à m

- La tâche ne s'exécute pas sur un autre machine que celle prévue

(xik <= M * Uik) ou mettre la grosse disjonction d'en haut ici aussi

T_i, x_i >= 0