

Ordonnancement et équité

Rapport de stage



Master d'informatique M2

Faculté de Sciences et Ingénierie de Sorbonne Université

Marion Caumartin

6 mars 2019

Table des matières

1	Introduction	1
2	Définition du problème	2
2.1	Ordonnancement	2
2.2	Equité	3
2.2.1	Définition	3
2.2.2	Propriétés	3

1 Introduction

Les problèmes d'ordonnancement constituent un domaine important en recherche opérationnelle. Ils traitent de l'affectation de tâches à des machines, et de l'exécution de ces tâches au cours du temps : il s'agit de savoir où - sur quelle machine - et quand commencera chaque tâche. Récemment, des problèmes d'ordonnancement en présence de différents acteurs ont été étudiés. Cependant, très peu de critères d'équité ont été appliqués aux problèmes d'ordonnancement (hormis le critère classique qui consiste à minimiser le coût maximal d'une machine).

La théorie du partage équitable est un domaine central en choix social computationnel. Elle s'intéresse à définir des règles d'affectation de ressources à des agents de manière à satisfaire au mieux les agents, sachant que les préférences de ceux-ci peuvent avoir différentes structures inhérentes au problème. Un concept fréquemment utilisé pour évaluer les affectations est celui de l'absence d'envie (envy freeness) : une affectation de ressources est dite sans-envie si aucun des agents ne préfère le lot de ressources d'un autre agent à son propre lot.

Le but de ce stage est d'appliquer des concepts et techniques développés en théorie du partage équitable, au domaine de l'ordonnancement. On cherche ainsi à évaluer la qualité des ordonnancements du point de vue de l'équité et à calculer des ordonnancements "équitables".

2 Définition du problème

2.1 Ordonnancement

On note m le nombre de machines.

On note s_i^j le slot de temps i de la machine j .

$\{1, \dots, n\}$ un ensemble d'agents.

Chaque agent k a un ensemble de $x \in \mathbb{N}$ tâches à réaliser et un critère f_k à minimiser.

On note J_i^k la tâche i de l'agent k .

Chaque tâche J_i^k a une durée p_i^k , peut avoir une deadline d_i^k et une date de disponibilité r_i^k .

Dans un premier temps, on considère que $\forall k \in \{1, \dots, n\}, \forall i \in \{1, \dots, x\}, p_i^k = 1$.

Un ordonnancement σ est défini par les dates de début des tâches : $\sigma(J_i^k)$. On note \mathcal{S} l'ensemble des ordonnancements valides. On note $\sigma(k)$ l'ordonnancement de l'agent k pour l'ordonnancement σ .

On note C_i^k la date de fin de la tâche i de l'agent k : $C_i^k = \sigma(J_i^k) + p_i^k$.

$C_{max}^k = \max_i C_i^k$: date de fin maximum (makespan)

$T_i^k = \max(0, C_i^k - d_i^k)$: retard

$L_i^k = C_i^k - d_i^k$: retard algébrique

$U_i^k = \begin{cases} 1 & \text{si } C_i^k > d_i^k \\ 0 & \text{sinon} \end{cases}$: vaut 1 si J_i^k est en retard, 0 sinon

$E_i^k = \max(0, d_i^k - C_i^k)$: avance

$S_i^k = (L_i^k)^2$

$D_i^k = |D_i^k|$

2.2 Équité

2.2.1 Définition

On cherche un ordonnancement où chaque agent a exactement x slot : $\forall k \in \{1, \dots, n\}, |\sigma(k)| = x$.

$$\forall k, k' \in \{1, \dots, n\}, \sigma(k) \cap \sigma(k') = \emptyset.$$

Les slots ne peuvent pas être divisés.

On note $f = (f_1, \dots, f_n)$.

Pareto-optimalité : σ domine σ' au sens de Pareto ssi

$$\begin{cases} \forall k \in \{1, \dots, n\}, f_k(\sigma(k)) \leq f_k(\sigma'(k)) \\ \exists k \in \{1, \dots, n\}, f_k(\sigma(k)) < f_k(\sigma'(k)) \end{cases}$$

utilitarian social welfare : minimiser $\sum_{k=1}^n f_k$.

egalitarian social welfare : minimiser $\max_{k=1}^n f_k$.

$$\text{leximin : } \sigma \succ_{lex} \sigma' \iff \exists i \in \{1, \dots, n-1\}, \begin{cases} \forall k < i, f_k(\sigma(k)) = f_k(\sigma'(k)) \\ f_i(\sigma(i)) < f_i(\sigma'(i)) \end{cases}$$

proportional fair share : on note f_k^* le coût si k disposait des machines pour lui seul.

$$\text{pfs}(k) = n f_k^*.$$

maxmin fair share : $\text{mfs}(k) = \min_{\sigma \in \mathcal{S}} \max_{i=1}^n f_k(\sigma(i))$.

Ordonnancement sans envie : $\forall k, j \in \{1, \dots, n\}, f_k(\sigma(k)) \leq f_k(\sigma(j))$

Envie entre deux agents : $e_{ij} = \max(0, f_i(\sigma(i)) - f_i(\sigma(j)))$

degré d'envie : $e_i = \max_j e_{ij}$ ou $e_i = \sum_j e_{ij}$

envie de la société : $\max_i e_i$ ou $\sum_i e_i$

Envie up to one slot : i envie j up to one slot ssi $\exists s \in \sigma(i), f_i(\sigma(i) \setminus s) \leq f_i(\sigma(j))$

Envie up to any slot : i envie j up to any slot ssi $\forall s \in \sigma(i), f_i(\sigma(i) \setminus s) \leq f_i(\sigma(j))$

2.2.2 Propriétés

Propriété 1. Au moins un ordonnancement egalitarian-optimal est Pareto-optimal.

Démonstration. Par l'absurde :

Soit σ un ordonnancement egalitarian-optimal. Supposons que σ n'est pas Pareto-optimal. Il existe donc un ordonnancement σ' Pareto-optimal qui domine σ et qui n'est pas egalitarian-optimal.

2 cas possibles :

- $\max_{i=1}^n f_i(\sigma'(i)) = \max_{i=1}^n f_i(\sigma(i))$: comme σ est egalitarian-optimal, alors σ' l'est aussi \implies contradiction.
- $\max_{i=1}^n f_i(\sigma'(i)) \leq \max_{i=1}^n f_i(\sigma(i))$: donc σ n'est pas egalitarian-optimal \implies contradiction. \square

Propriété 2. Un ordonnancement sans envie n'est pas toujours Pareto-optimal.

Exemple 1. On considère une instance avec une machine et deux agents A et B qui veulent optimiser le même critère. Chaque agent a deux tâches à exécuter sur la machine. On note A_i (resp. B_i) la tâche i de l'agent A (resp. B) pour $i \in \{1, 2\}$. Chaque tâche a une deadline : $d_1^A = 1$, $d_2^A = 3$, $d_1^B = 2$ et $d_2^B = 4$.

On appelle σ_1 l'ordonnancement suivant :

A_1	B_1	B_2	A_2
-------	-------	-------	-------

On appelle σ_2 l'ordonnancement suivant :

A_1	B_1	A_2	B_2
-------	-------	-------	-------

- Si A et B veulent minimiser $\sum T_i$: σ_1 est sans-envie mais n'est pas optimal. En effet, $\begin{cases} f_A(\sigma_1(A)) = 1 \leq f_A(\sigma_1(B)) = 1 \\ f_B(\sigma_1(B)) = 0 \leq f_B(\sigma_1(A)) = 0 \end{cases}$ et $\begin{cases} f_A(\sigma_2(A)) = 0 < f_A(\sigma_1(A)) = 1 \\ f_B(\sigma_2(B)) = 0 \leq f_B(\sigma_1(B)) = 0 \end{cases}$
- Si A et B veulent minimiser $\sum U_i$: σ_1 est sans-envie mais n'est pas optimal. En effet, $\begin{cases} f_A(\sigma_1(A)) = 1 \leq f_A(\sigma_1(B)) = 1 \\ f_B(\sigma_1(B)) = 0 \leq f_B(\sigma_1(A)) = 0 \end{cases}$ et $\begin{cases} f_A(\sigma_2(A)) = 0 < f_A(\sigma_1(A)) = 1 \\ f_B(\sigma_2(B)) = 0 \leq f_B(\sigma_1(B)) = 0 \end{cases}$
- Si A et B veulent minimiser $\sum E_i$: σ_1 est sans-envie mais n'est pas optimal. En effet, $\begin{cases} f_A(\sigma_1(A)) = 0 \leq f_A(\sigma_1(B)) = 0 \\ f_B(\sigma_1(B)) = 1 \leq f_B(\sigma_1(A)) = 1 \end{cases}$ et $\begin{cases} f_A(\sigma_2(A)) = 0 \leq f_A(\sigma_1(A)) = 0 \\ f_B(\sigma_2(B)) = 0 < f_B(\sigma_1(B)) = 1 \end{cases}$
- Si A et B veulent minimiser $\sum S_i$: σ_1 est sans-envie mais n'est pas optimal. En effet, $\begin{cases} f_A(\sigma_1(A)) = 1 \leq f_A(\sigma_1(B)) = 1 \\ f_B(\sigma_1(B)) = 1 \leq f_B(\sigma_1(A)) = 1 \end{cases}$ et $\begin{cases} f_A(\sigma_2(A)) = 0 < f_A(\sigma_1(A)) = 1 \\ f_B(\sigma_2(B)) = 0 < f_B(\sigma_1(B)) = 1 \end{cases}$
- Si A et B veulent minimiser $\sum D_i$: σ_1 est sans-envie mais n'est pas optimal. En effet, $\begin{cases} f_A(\sigma_1(A)) = 1 \leq f_A(\sigma_1(B)) = 1 \\ f_B(\sigma_1(B)) = 1 \leq f_B(\sigma_1(A)) = 1 \end{cases}$ et $\begin{cases} f_A(\sigma_2(A)) = 0 < f_A(\sigma_1(A)) = 1 \\ f_B(\sigma_2(B)) = 0 < f_B(\sigma_1(B)) = 1 \end{cases}$

Propriété 3. Minimiser $\sum L_i$ revient à minimiser $\sum C_i$.

Démonstration. $\sum L_i = \sum (C_i - d_i) = \sum C_i - \sum d_i$. Or $\sum d_i$ est une constante. Donc si on minimise $\sum C_i$, on minimise également $\sum L_i$. \square

Exemple 2. On considère une instance avec 2 machines, 3 agents A , B et D qui veulent optimiser $\sum C_i$. Chaque agent a 3 tâches à exécuter. On note A_i (resp. B_i et D_i) la tâche i de l'agent A (resp. B et D) pour $i \in \{1, 2, 3\}$.

On appelle σ_1 l'ordonnancement suivant :

A_1	D_1		A_2	B_3
B_1	D_3	B_2	A_3	D_2

On appelle σ_2 l'ordonnancement suivant :

A_1	D_1	D_2	D_3	B_3
B_1	B_2	A_2	A_3	

σ_1 est sans-envie mais n'est pas optimal. En effet,

$$\left\{ \begin{array}{l} f_A(\sigma_1(A)) = 9 \leq f_A(\sigma_1(B)) = 9 \\ f_A(\sigma_1(A)) = 9 \leq f_A(\sigma_1(D)) = 9 \end{array} \right\}, \left\{ \begin{array}{l} f_B(\sigma_1(B)) = 9 \leq f_B(\sigma_1(A)) = 9 \\ f_B(\sigma_1(B)) = 9 \leq f_B(\sigma_1(D)) = 9 \end{array} \right\},$$

$$\left\{ \begin{array}{l} f_D(\sigma_1(D)) = 9 \leq f_D(\sigma_1(A)) = 9 \\ f_D(\sigma_1(D)) = 9 \leq f_D(\sigma_1(B)) = 9 \end{array} \right\} \text{ et } \left\{ \begin{array}{l} f_A(\sigma_2(A)) = 8 < f_A(\sigma_1(A)) = 9 \\ f_B(\sigma_2(B)) = 9 \leq f_B(\sigma_1(B)) = 9 \\ f_D(\sigma_2(D)) = 8 < f_D(\sigma_1(D)) = 9 \end{array} \right\}.$$