

Algorithme approché probabiliste pour le problème $Q||\sum_j w_j C_j$ en tenant compte de l'énergie consommée

Eric Angel¹, Evripidis Bampis², Fadi Kacem¹

¹ IBISC EA 4526 ; Université d'Evry Val d'Essonne ; Boulevard Mitterrand, 91025 Evry, France
{angel, fkacem}@ibisc.fr

² LIP6-CNRS UMR 7606 ; Université Pierre et Marie Curie ; 4 place Jussieu 75005 Paris, France
Evripidis.Bampis@lip6.fr

Mots-clés : *Ordonnancement, énergie, relaxation linéaire, algorithme probabiliste, machines uniformes.*

Présentation du problème. On considère le problème d'ordonnancement $Q||\sum_j w_j C_j$. Soit un ensemble \mathcal{J} de n tâches, ainsi que m machines uniformes. Chaque tâche j est caractérisée par une quantité de travail p_j , et chaque machine i par une vitesse v_i . La durée d'exécution d'une tâche dépend de la machine sur laquelle elle est exécutée : si la tâche j s'exécute sur la machine i alors son temps d'exécution est $p_{ij} = p_j/v_i$. La préemption et la migration ne sont pas autorisées, i.e chaque tâche doit être exécutée en totalité sur une seule machine sans interruption. De plus, une machine ne peut exécuter qu'une seule tâche à un instant donné. À chaque tâche j , on associe un poids $w_j \geq 0$ et on note par C_j sa date de fin dans un ordonnancement. Les problématiques de minimisation d'énergie prennent une place de plus en plus importante (voir par exemple le survey [1]). Afin de tenir compte de l'énergie consommée, nous adoptons le modèle de Yao et al. [3] qui est devenu standard. Si une machine tourne à la vitesse v pendant t unités de temps, alors elle dépense une énergie (proportionnelle à) $v^\alpha t$, avec $\alpha > 1$ une constante (souvent égale à 2 ou 3). Étant donnée une borne supérieure, $E \geq 0$, sur l'énergie consommée, le problème consiste donc à trouver un ordonnancement réalisable qui minimise $\sum_{j \in \mathcal{J}} w_j C_j$ sans que l'énergie consommée ne dépasse la borne E . Ce problème est NP-difficile étant donnée que le même problème sans la contrainte sur l'énergie est NP-difficile.

Algorithme probabiliste. L'algorithme probabiliste repose sur l'idée de relaxer le problème en utilisant la programmation linéaire, puis de faire appel à une interprétation probabiliste de la solution optimale du programme linéaire (PL) pour réaliser une affectation aléatoire des tâches sur les machines dans le problème initial.

Soit $T = \sum_{j \in \mathcal{J}} \max_i p_{ij}$ l'horizon du temps. On discrétise l'intervalle $[0, T]$ en des intervalles de tailles géométriques. Pour un réel $\eta > 0$ fixé, soit L le plus petit entier tel que $(1 + \eta)^L \geq T$. Ainsi, on définit l'ensemble des intervalles I_l , $0 \leq l \leq L$, comme suit :

$$I_l = \begin{cases} [0, 1] & \text{si } l = 0, \\ [(1 + \eta)^{l-1}, (1 + \eta)^l] & \text{sinon.} \end{cases}$$

Pour $0 \leq l \leq L$, on note $|I_l|$ la taille de l'intervalle I_l . Nous introduisons les variables y_{ijl} pour tout $1 \leq i \leq m$, $j \in \mathcal{J}$ et $0 \leq l \leq L$, où $y_{ijl} \cdot |I_l|$ est le temps d'exécution de la tâche j sur la machine i pendant l'intervalle I_l . Nous adoptons aussi la convention suivante, $(1 + \eta)^{l-1} = \frac{1}{2}$ pour $l = 0$. Soit le programme linéaire suivant :

$$\begin{aligned}
\min \sum_{j \in \mathcal{J}} w_j C_j^{LP} \\
\text{s.c. } \sum_{i=1}^m \sum_{l=0}^L \frac{y_{ijl} \cdot |I_l|}{p_{ij}} = 1, & \quad \forall j \in \mathcal{J}, \quad (1) \\
\sum_{j \in \mathcal{J}} y_{ijl} \leq 1, & \quad \forall i = 1 \dots m, \text{ et } \forall l = 0 \dots L, \quad (2) \\
C_j^{LP} \geq \sum_{i=1}^m \sum_{l=0}^L \left(\frac{y_{ijl} |I_l|}{p_{ij}} \cdot (1 + \eta)^{l-1} + \frac{1}{2} y_{ijl} \cdot |I_l| \right), & \quad \forall j \in \mathcal{J}, \quad (3) \\
\sum_{i=1}^m \sum_{j \in \mathcal{J}} \sum_{l=0}^L v_i^\alpha \cdot y_{ijl} \cdot |I_l| \leq E, & \quad (4) \\
y_{ijl} \geq 0, & \quad \forall i = 1 \dots m, \forall j \in \mathcal{J}, \text{ et } \forall l = 0 \dots L, \quad (5) \\
C_j^{LP} \geq 0, & \quad \forall j \in \mathcal{J}. \quad (6)
\end{aligned}$$

Le PL ci-dessus est inspiré d'un PL présenté dans [2] pour résoudre le problème $R|r_j| \sum w_j C_j$ avec des tâches de durées d'exécution entières. Nous avons montré que l'inégalité (3) restait valide dans notre cas, bien que les durées p_{ij} sont en général fractionnaires. Plus précisément, il est possible de montrer que la partie droite de la contrainte (3) est une borne inférieure sur date de fin dans toute solution réalisable si les durées d'exécution vérifient : $p_{ij} \geq 1$ pour toute tâche $j \in \mathcal{J}$.

Pour les instances dans lesquelles il existe des tâches avec des durées d'exécution inférieures à 1, il suffit de normaliser les quantités de travail p_j pour que la plus petite durée d'exécution soit supérieure à 1 et de multiplier la borne sur l'énergie E par le facteur de normalisation. Il est facile de voir que ce PL constitue une relaxation de notre problème. Notons que le fait d'utiliser des intervalles géométriques garantit un nombre de variables polynomial en fonction de la taille de l'instance.

L'algorithme consiste à calculer une solution optimale y_{ijl} du PL, puis à affecter, de façon indépendante, chaque tâche j à une paire machine-intervalle (i, I_l) avec la probabilité $\frac{y_{ijl} \cdot |I_l|}{p_{ij}}$ et à poser $t_j = (1 + \eta)^{l-1}$. Finalement, les tâches affectées à chaque machine sont ordonnancées suivant l'ordre croissant des t_j .

Théorème : Pour chaque couple de valeurs positives (a, b) vérifiant $\frac{1}{a} + \frac{1}{b} \leq 1$, cet algorithme construit, avec une probabilité supérieure ou égale à $1 - \frac{1}{a} - \frac{1}{b}$, un ordonnancement σ tel que $\sum_j w_j C_j(\sigma) < 2a(1 + \eta) OPT(E)$ et tel que l'énergie consommée par cet ordonnancement soit inférieure à bE , avec $OPT(E)$ la somme pondérée des dates de fin dans un ordonnancement optimal consommant une énergie d'au plus E .

Il est possible de généraliser ce résultat en supposant que la vitesse d'exécution d'une tâche est fonction de la machine sur laquelle elle est exécutée.

Références

- [1] S. Albers. *Energy efficient algorithms*. Communications of the ACM 53(5), 86–96, 2010.
- [2] A.S. Schulz, M. Skutella. *Scheduling unrelated machines by randomized rounding*. SIAM Journal on Discrete Mathematics 15(4), 450–469, 2002.
- [3] F. Yao, A. Demers, S. Shenker. *A scheduling model for reduced CPU energy*. FOCS'95, 374–382.