Maximisation de profit dans les systèmes informatiques sous contraintes énergétiques

Vincent Chau, Eric Angel, Evripidis Bampis

Maximisation de profit dans les systèmes informatiques sous contraintes énergétiques *

Vincent Chau † Eric Angel † Evripidis Bampis ‡

Résumé

L'économie d'énergie dans les systèmes informatiques est aujourd'hui un sujet très important aussi bien du point de vue écologique qu'économique. Dans ce contexte, on considère le problème d'ordonnancement suivant : étant donné un ensemble de tâches et un processeur qui peut varier sa vitesse dynamiquement, l'objectif est d'exécuter un maximum de tâches sans dépasser le budget sur l'énergie. Chaque tâche est caractérisée par sa quantité de travail, sa date de relâchement et sa date d'échéance. Bien que le problème de la minimisation d'énergie ait été résolue en temps polynomial [Yao et al., FOCS'95], la complexité du problème de la maximisation du nombre de tâches exécutées reste ouverte. On répond partiellement à cette question en proposant un algorithme de programmation dynamique qui résout le problème en temps pseudo-polynomial. Notre algorithme peut être adapté pour le cas pondéré dans lequel chaque tâche est associée à un poids et où l'objectif est de maximiser la somme des poids des tâches exécutées.

1 Introduction

Le problème d'ordonnancement dans lequel on veut minimiser la consommation d'énergie sachant que le processeur peut varier sa vitesse a été initialement analysé par Yao, Demers et Shenker [1]. Plus formellement, on veut ordonnancer un ensemble de n tâches tel que chaque tâche J_j est définie selon une date de relâchement r_j, une date d'échéance d_j et une quantité de travail p_j sur un processeur dont la vitesse peut être ajustée dynamiquement. L'objectif est de produire un ordonnancement de toutes les tâches tel que la consommation d'énergie soit minimale.

Le processeur peut exécuter au plus une tâche à la fois. La vitesse doit être positive ou nulle. On mesure la vitesse du processeur en unités de travail effectué par unité de temps. Si $s(t)$ est la vitesse du processeur à l'instant t, alors le total de travail effectué dans l'intervalle de temps $[t, t')$ est égal à $\int_t^{t'} s(u)du$. De plus, on suppose que la consommation d'énergie du processeur est une fonction convexe par rapport à la vitesse. En particulier, pour un instant t donné, la consommation énergétique du processeur est $P(t) = s(t)^\alpha$, où $\alpha > 1$ est une constante. On note également que si le processeur est à vitesse constante s durant l'intervalle $[t, t')$, alors il effectue $(t' - t) \times s$ unités de travail et la consommation d'énergie est égale à $(t' - t) \times s^\alpha$. Chaque tâche J_j peut être uniquement exécutée durant la période $[r_j, d_j)$. De plus, on autorise la préemption des tâches, c'est-à-dire qu'une tâche peut être interrompue et être reprise plus tard. Dans ce papier, on considère le problème de maximisation du nombre de tâches exécutées sans dépasser un budget d'énergie E. La maximisation de profit dans ce contexte est très naturelle depuis la généralisation des appareils portables puisqu'ils possèdent une capacité d'énergie limitée. Différentes variantes de ce problème dans le cadre en-ligne ont été étudiées dans la littérature, mais la complexité du problème dans le cadre hors-ligne n'est pas connue. On détaillera ce problème par la suite.

1.1 Travaux liés et contribution

Angel et al. [2] ont été les premiers à considérer le problème de maximisation de profit sous contrainte d'énergie dans le cas hors-ligne. Ils ont étudié une famille d'instances particulière dans laquelle pour chaque paire de tâches J_i et J_j, on a $r_i \leq r_j$, si et seulement si, $d_i \leq d_j$ et ils ont proposé un algorithme polynomial. Plus récemment, Antoniadis et al. [3] considèrent une généralisation du problème du sac-à-dos où l'objectif est de maximiser la somme des profits des objets choisis moins un coût en fonction de la somme des poids de ces objets. Lorsque la fonction de coût est convexe, le problème considéré est

*Ce travail s'inscrit dans le contexte du projet ANR TODO (09-EMER-010) et PHC CAI YUANPEI (27927VE).

†IBISC, Université d'Evry, {Eric.Angel, Vincent.Chau}@ibisc.univ-evry.fr

‡Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France, Evripidis.Bampis@lip6.fr

similaire. En effet, chaque objet devient une tâche, et la fonction de coût convexe correspond à la fonction de la consommation d'énergie. Antoniadis et al. proposent un FPTAS et une 2-approximation pour le cas où les tâches n'ont ni date de relâchement, ni date d'échéance.

Cependant, la complexité du problème général reste ouverte. Dans ce papier, on montre qu'il existe un algorithme pseudo-polynomial pour résoudre le problème général. Pour la version pondérée, i.e. la version où les tâches sont associées en plus à un poids et où on veut maximiser la somme de poids total des tâches exécutées, le problème est $\mathcal{NP}$-difficile même si toutes les tâches ont la même date de relâchement et la même date d'échéance. Notre algorithme peut être généralisé pour le cas pondéré.

2 Préliminaires

Parmi tous les ordonnancements ayant un même profit, on cherche celui qui a la consommation d'énergie minimale. Alors, étant donné un ensemble de tâches $J^* \subseteq J$ d'un ordonnancement optimal pour le problème de la maximisation de profit, on peut simplement appliquer l'algorithme optimal pour le problème de la minimisation de la consommation d'énergie [1] sur cet ensemble de tâches. En se basant sur cette observation, on peut utiliser quelques propriétés d'un ordonnancement de consommation d'énergie minimale dans notre analyse.

Soit $t_1, t_2, \ldots, t_K$ les dates qui correspondent soit à une date de relâchement, soit à une date d'échéance. On numérote les valeurs de t_i dans l'ordre croissant, c'est-à-dire $t_1 < t_2 < \ldots < t_K$. Le théorème suivant provient de [1] et a été prouvé dans [4].

Théorème 1. *Un ordonnancement réalisable pour le problème de la minimisation de la consommation d'énergie est optimal si et seulement si les propriétés suivantes sont vérifiées :*

1. *Chaque tâche J_j est exécutée à une vitesse constante s_j.*
2. *Le processeur n'est pas en état inactif pour tout $t \in (r_j, d_j]$ pour toute tâche $J_j \in J$.*
3. *Le processeur a une vitesse constante durant tous les intervalles $(t_i, t_{i+1}]$, pour $1 \leq i \leq K-1$.*
4. *Si d'autres tâches sont exécutées durant $[r_j, d_j]$, alors leur vitesse est nécessairement supérieure ou égale à celle de la tâche J_j.*

Tous les ordonnancements considérés par la suite respecteront le Théorème 1 pour avoir la consommation d'énergie minimale quelque soit l'ensemble de tâches choisies. On supposera également que les tâches sont triées dans l'ordre croissant de leur date d'échéance (ordre EDF (Earliest Deadline First)), i.e. $d_1 \leq d_2 \leq \ldots \leq d_n$. De plus, on suppose que les dates de relâchement, les dates d'échéance et les quantités de travail sont des valeurs entières.

Définition 1. *Soit $J(k, s, t) = \{J_j \mid j \leq k \text{ et } s \leq r_j < t\}$ l'ensemble des tâches, parmi les k premières tâches selon l'ordre* EDF*, dont les dates de relâchement sont entre s et t.*

Lemme 1. *La durée totale durant laquelle le processeur fonctionne à une même vitesse dans une solution optimale est entière.*

Démonstration. La durée totale d'une même vitesse est définie par un ensemble d'intervalles $(t_i, t_{i+1}]$ pour $1 \leq i \leq K-1$ grâce à la propriété 3 du Théorème 1. Puisque chaque t_i correspond à une date de relâchement ou à une date d'échéance, alors $t_i \in \mathbb{N},\ 1 \leq i \leq K$. Ainsi, la durée totale durant laquelle le processeur fonctionne à la même vitesse dans une solution optimale est bien une valeur entière. □

Définition 2. *Soit $H = \max_j d_j - \min_j r_j$ l'horizon de temps d'un ordonnancement quelconque de l'instance. Pour simplifier les notations, on suppose que $\min_j r_j = 0$.*

Définition 3. *Soit $P = \sum_j p_j$ la somme totale des quantités de travail de toutes les tâches.*

Définition 4. *On appelle un ordonnancement* EDF *(Earliest Deadline First), un ordonnancement dans lequel à chaque instant, le processeur exécute la tâche qui a la plus petite date d'échéance parmi les tâches disponibles.*

Par la suite, tous les ordonnancements considérés sont des ordonnancements EDF.

3 La programmation dynamique

Dans cette partie, on propose un algorithme optimal basé sur la programmation dynamique qui dépend de la longueur de l'horizon de temps et du total de quantités de travail P. Comme mentionné précédemment, parmi les ordonnancement ayant un même profit, notre algorithme construit un ordonnancement de consommation d'énergie minimale. Pour un sous-ensemble de tâches $S \subseteq J$, on appelle un ordonnancement S s'il ne contient que les tâches de S.

Définition 5. *Soit $G_k(s,t,u)$ la consommation d'énergie minimale d'un ordonnancement S avec $S \subseteq J(k,s,t)$ tel que $|S| = u$ et tel que les tâches de S sont entièrement exécutées dans $[s,t]$.*

Etant donné un budget sur l'énergie E qu'on ne peut pas dépasser, la fonction objectif est $\max\{u \mid G_n(0,d_{max},u) \leq E;\ 0 \leq u \leq n\}$.

Définition 6. *Soit $F_{k-1}(x,y,u,\ell,i,a,h)$ la consommation d'énergie minimale d'un ordonnancement S avec $S \subseteq J(k-1,x,y)$ tel que $|S| = u$ et tel que les tâches dans S sont entièrement exécutées dans $[x,y]$ durant au plus $a + h \times \frac{\ell}{i}$ unités de temps. De plus, on suppose que chaque bloc maximal de tâches consécutives de S commence à une date de relâchement et a une longueur égale à $a' + h' \times \frac{\ell}{i}$ avec $a', h' \in \mathbb{N}$.*

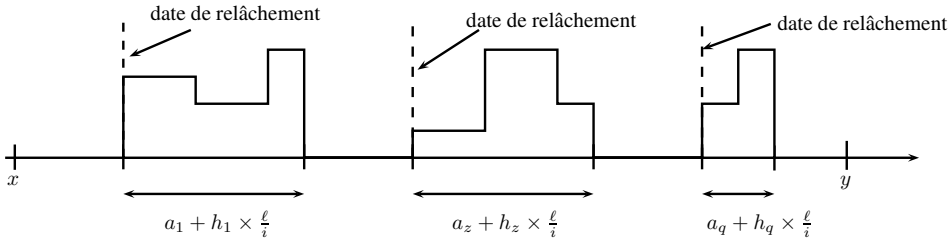

FIGURE 1 – Illustration de $F_{k-1}(x,y,u,\ell,i,a,h)$ dans la Définition 6 tel que $\sum_{z=1}^{q} a_z + h_z \times \frac{\ell}{i} = a + h \times \frac{\ell}{i}$

Ensuite, on définit l'ensemble des dates importantes d'un ordonnancement optimal dans lequel chaque tâche peut commencer et on montre que la taille de cet ensemble est pseudo-polynomiale.

Définition 7. *Soit $\Phi = \{s + h \times \frac{\ell}{i} \leq H \mid i = 1,\dots,P;\ h = 0,\dots,i;\ s = 0,\dots,H;\ \ell = 1,\dots,H\}$.*

Proposition 1. *Il existe un ordonnancement optimal $\mathcal{O}$ dans lequel, pour chaque tâche, sa date de début et sa date de fin est dans l'ensemble Φ, et tel que chaque tâche est entièrement exécutée avec une vitesse $\frac{i}{\ell}$ pour $i = 1,\dots,P$ et $\ell = 1,\dots,H$.*

L'idée de la décomposition est de trouver la date de début x et la date de fin y de la tâche J_k qui n'est pas encore exécutée (voir Figure 2). On obtient donc trois sous-ordonnancement définis sur les intervalles $[s,x]$, $[x,y]$ et $[y,t]$ qui sont indépendants. Le premier et le troisième sous-ordonnancements sont connus à cette étape puisqu'ils ne doivent pas contenir la tâche J_k. Il faut donc réserver de l'espace libre dans l'intervalle $[x,y]$ pour la tâche J_k. On sait que les différentes vitesses possibles pour chaque tâche sont $\frac{i}{\ell}$ pour $i = 1,\dots,P$ et $\ell = 1,\dots,H$ grâce à la Proposition 1. En fixant la vitesse de la tâche J_k à $\frac{i}{\ell}$, on sait que son temps d'exécution sera $p_k \frac{i}{\ell}$. Le deuxième sous-ordonnancement doit être défini sur au maximum $y - x - p_k \ell / i$ unités de temps pour permettre d'exécuter entièrement la tâche J_k dans l'intervalle $[x,y]$ à la vitesse souhaitée.

Proposition 2.

$$G_k(s,t,u) = \min \begin{cases} G_{k-1}(s,t,u) \\ \min\limits_{\substack{x \in \Phi \\ 0 \leq u_1 \leq u \\ 0 \leq u_2 \leq u \\ 0 \leq u_1 + u_2 \leq u-1 \\ 0 \leq a \leq H;\ 1 \leq \ell \leq H \\ 1 \leq i \leq P;\ 0 \leq h \leq P \\ y - x = a + (p_k + h)\frac{\ell}{i} \\ r_k \leq x \leq y \leq d_k}} \left\{ \begin{array}{l} G_{k-1}(s,x,u_1) + F_{k-1}(x,y,u_2,\ell,i,a,h) \\ + \left(\frac{i}{\ell}\right)^{\alpha-1} p_k + G_{k-1}(y,t,u-u_1-u_2-1) \end{array} \right\} \end{cases}$$

$G_0(s,t,0) = 0\ \forall s,t \in \Phi$
$G_0(s,t,u) = +\infty\ \forall s,t \in \Phi$ *et* $u > 0$

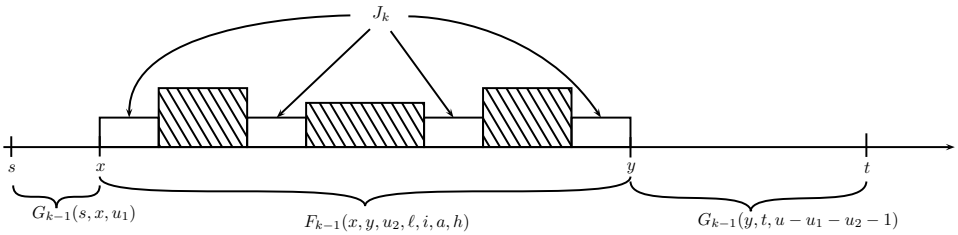

FIGURE 2 – Illustration de la Proposition 2 où x est la première date de début de J_k et y est la dernière date de fin de J_k

Proposition 3.

$$F_{k-1}(x, y, u, \ell, i, a, h) = \min_{\substack{0 \le a' \le a;\ 0 \le h' \le h \\ x \le x' = r_j \le y;\ j \le k \\ 1 \le \beta \le u \\ y' = x' + a' + h' \times \frac{\ell}{i} \le y}} \{G_{k-1}(x', y', \beta) + F_{k-1}(y', y, u - \beta, \ell, i, a - a', h - h')\}$$

$F_{k-1}(x, y, 0, \ell, i, a, h) = 0$
$F_{k-1}(x, y, u, \ell, i, 0, 0) = +\infty$

L'ordonnancement correspondant à $F_{k-1}(x, y, u, \ell, i, a, h)$ peut être effectivement décomposé de la sorte. D'après la Définition 6, chaque bloc de tâches commence à une date de relâchement et la longueur de chaque bloc est de la forme $a' + h' \times \frac{\ell}{i}$. Au final, l'ordonnancement correspondant à $F_{k-1}(x, y, u, \ell, i, a, h)$ est un ensemble de $G_{k-1}(x', y', .)$ représenté par des blocs hachurés dans la Figure 2.

Théorème 2. *Le problème de maximisation de tâches peut être résolu en temps $O(n^6 H^9 P^9)$ et avec une quantité de mémoire en $O(nH^6 P^6)$.*

Le programme dynamique peut être adapté pour le cas pondéré du problème et a une complexité en temps égale à $O(n^2 W^4 H^9 P^9)$ où W est la somme totale de tous les poids. Pour ce faire, on modifie les deux tables de la programmation dynamique en considérant le poids total des tâches exécutées au lieu de considérer le nombre de tâches exécutées.

4 Conclusion

Dans ce papier, on montre qu'il existe un algorithme optimal en temps pseudo-polynomial pour le problème de la maximisation de tâches sous contrainte d'énergie dans le cadre hors-ligne. Ce résultat est une première réponse concernant la complexité du problème qui montre qu'il n'est pas $\mathcal{NP}$-difficile au sens fort, mais la question de s'il existe un algorithme polynomial pour ce problème reste encore ouverte.

Références

[1] F. Frances Yao, Alan J. Demers, and Scott Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382. IEEE Computer Society, 1995.

[2] Eric Angel, Evripidis Bampis, Vincent Chau, and Dimitrios Letsios. Throughput maximization for speed-scaling with agreeable deadlines. In T.-H. Hubert Chan, Lap Chi Lau, and Luca Trevisan, editors, *TAMC*, volume 7876 of *Lecture Notes in Computer Science*, pages 10–19. Springer, 2013.

[3] Antonios Antoniadis, Chien-Chung Huang, Sebastian Ott, and José Verschae. How to pack your items when you have to buy your knapsack. In Krishnendu Chatterjee and Jiri Sgall, editors, *MFCS*, volume 8087 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2013.

[4] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.