Programmation linéaire mixte et programmation par contraintes pour un problème d'ordonnancement a contraintes énergétiques

Margaux Nattaf, Christian Artigues, Pierre Lopez

Programmation linéaire mixte et programmation par contraintes pour un problème d'ordonnancement à contraintes énergétiques

Margaux Nattaf, Christian Artigues, and Pierre Lopez

LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{nattaf,artigues,lopez}@laas.fr

November 3, 2016

Abstract

Nous considérerons un problème d'ordonnancement cumulatif où la durée des tâches ainsi que leur profil de consommation de ressource ne sont pas fixées. Ce profil, qui peut varier en fonction du temps, est une variable de décision du problème dont dépend la durée de la tâche associée. Dans le cas où le profil de consommation d'une tâche ne peut varier que pour des temps discret, nous présentons un modèle de programmation par contraintes et un modèle de programmation linéaire en nombres entiers (PLNE). De plus, deux algorithmes de propagation sont décrits et des inégalités valides déduites de la programmation par contraintes viennent renforcer le PLNE. Ces modèles sont ensuite comparés par le biais d'expérimentations.

1 Introduction

Nous étudions un problème d'ordonnancement avec ressource continue et contraintes énergétiques, le Continuous Energy-Constrained Scheduling Problem (CECSP). Dans ce problème, un ensemble de tâches $\mathcal{A} = \{1, \ldots, n\}$ utilisant une ressource continue et cumulative de capacité limitée B doit être ordonnancer. La quantité de ressource nécessaire à l'éxecution d'une tâche n'est pas fixée mais - le profil de consommation de cette dernière est une fonction $b_i(t)$ définie pour tout $t \in \mathbb{R}$[1] - doit être déterminé. Une fois la tâche commencée et jusqu'à sa date de fin, la fonction $b_i(t)$ doit être comprise entre une valeur maximale, b_i^{max}, et minimale, b_i^{min}.

De plus, la consommation, à un instant t, d'une partie de la ressource permet la production d'une certaine quantité d'énergie et, une tâche finit lorqu'elle a

[1] Le domaine de définition de la fonction peut être réduit mais, pour faciliter les notations, nous supposons qu'elle est définie pout tout $t \in \mathbb{R}$.

reçu une énergie W_i. Cette énergie est calculée par le biais d'une fonction de rendement f_i, propre à chaque tâche. Dans cet article, ces fonctions sont supposées continues, croissantes, affines et peuvent être exprimées de la manière suivante:

$$f_i(b) = \begin{cases} 0 & \text{si } b = 0 \\ a_i * b + c_i & \text{si } b_i^{min} = 0 \text{ et } b \in]b_i^{min}, b_i^{max}] \\ a_i * b + c_i & \text{si } b_i^{min} \neq 0 \text{ et } b \in [b_i^{min}, b_i^{max}] \end{cases}$$

avec $a_i > 0$ et $c_i \geq -a_i * b_i^{min}$ pour s'assurer que $f_i(b) \geq 0,\ \forall b \in [b_i^{min}, b_i^{max}]$.

Dans la suite, nous dénotons par $\underline{s_i}$ et $\overline{s_i}$ la date de début au plus tôt et au plus tard de i et par $\underline{e_i}$ et $\overline{e_i}$ la date de fin au plus tôt et au plus tard de i.

Pour trouver une solution pour le CECSP, nous devons déterminer, pour chaque tâche $i \in \mathcal{A}$, sa date de début s_i, sa date de fin e_i et sa fonction d'allocation de ressource $b_i(t)$, $\forall t \in \mathcal{T} = [\min_{i \in \mathcal{A}} \underline{s_i}, \max_{i \in \mathcal{A}} \overline{e}_i]$. De plus, ces variables doivent satisfaire les contraintes suivantes:

$$\underline{s_i} \leq s_i < e_i \leq \overline{e}_i \qquad \forall i \in \mathcal{A} \tag{1}$$

$$b_i^{min} \leq b_i(t) \leq b_i^{max} \qquad \forall i \in \mathcal{A},\ \forall t \in [s_i, e_i] \tag{2}$$

$$b_i(t) = 0 \qquad \forall i \in \mathcal{A},\ \forall t \notin [s_i, e_i] \tag{3}$$

$$\int_{s_i}^{e_i} f_i(b_i(t))dt = W_i \qquad \forall i \in \mathcal{A} \tag{4}$$

$$\sum_{i \in \mathcal{A}} b_i(t) \leq B \qquad \forall t \in \mathcal{T} \tag{5}$$

L'objectif auquel nous nous sommes intéressés est la minimisation de la consommation totale de la ressource. Dans [4], les auteurs montrent que trouver une solution admissible pour le CECSP est déjà un problème NP-complet.

De plus, une instance ayant des données seulement entières peut n'avoir que des solutions à valeurs dans $\mathbb{R}$ [4]. Cependant, une dilatation de l'instance, i.e. multiplier les données par un certain coefficient α, permet de palier à ce problème.

De ce fait et dans le but de résoudre des instances entières, nous nous sommes intéressés, dans un premier temps, à la version discrete du CECSP, le DECSP (Discrete Energy Constrained Scheduling Problem). Dans ce problème, toutes les données sont supposées entières et les domaines de chaque variable ne contiennent que des valeurs entières, i.e. $s_i,\ e_i,\ b_i(t) \in \mathbb{N}$ et $b_i(t)$ est défini $\forall t \in \mathcal{T}_\mathcal{D} = \{\min_{i \in \mathcal{A}} \underline{s_i}, \ldots, \max_{i \in \mathcal{A}} \overline{e}_i\}$.

Pour ce problème, nous présentons un modèle de programmation par contraintes (PPC) permettant l'utilisation des algorithmes de propagation mis en place pour la contrainte cumulative, nottament [3]. Un modèle de programmation linéaire en nombres entiers (PLNE) est aussi présenté. Ce modèle est ensuite renforcées à l'aide d'inégalités valides déduite du raisonnement énergétique [2].

Ces deux modèles sont ensuite testés sur des instances à données entières, avec et sans dilatation.

2 Modèle de programmation par contrainte

Dans cette section, nous présentons le modèle de programmation par contraintes défini pour le DECSP.

Pour modéliser le DECSP à l'aide de la PPC, nous divisons chaque tâche i en deux sous-tâches i_{min} et i_{preem}. La première, i_{min}, est une tâche ayant une consommation de ressource fixe, égale à b_i^{min}, et une durée variable p_i. Cette tâche représente la quantité de ressource obligatoirement consommée par une activité durant son exécution, i.e. b_i^{min}. La seconde, i_{preem} est une tâche préemptive optionnelle, consommant une quantité variable de ressource comprise entre 0 et $b_i^{max} - b_i^{min}$ et devant s'exécuter en même temps que i_{min}. Cette tâche est elle-même divisée en sous-tâches $i_{preem}^{\ell},\ \ell \in \{1, \ldots, e_i - s_i\} = \mathcal{L}_i$. Notons que $|\mathcal{L}_i| = e_i - s_i \leq \lceil \frac{W_i}{f_i(b_i^{min})} \rceil$.

Exemple 1. *Considérons la tâche possédant les attributs suivant:* 5 *définie de la manière suivante:* $\underline{s_i} = 0$, $\overline{e_i} = 6$, $W_i = 28$, $b^{min} = 1$, $b^{max} = 5$ *et* $f_i(b) = 2b + 1$. *La Figure 1 présente un ordonnancement de cette tâche (à droite) et l'ordonnancement correspondant donné par le modèle (à gauche) avec* $b_{i_{preem}^2} = 0$.

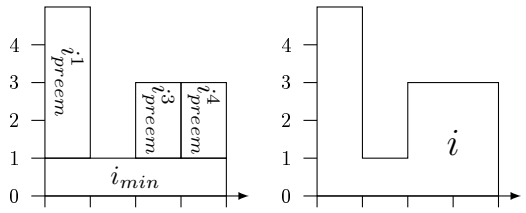

Figure 1: Exemple de solution du modèle PPC.

Le problème du DECSP peut alors être formulé à l'aide des variables:

- $i_{min} = \{s_{i_{min}}, e_{i_{min}}, b_{i_{min}}, p_{i_{min}}\},\ \forall i \in \mathcal{A}$
- $i_{preem}^{\ell} = \{s_{i_{preem}^{\ell}}, e_{i_{preem}^{\ell}}, b_{i_{preem}^{\ell}}, p_{i_{preem}^{\ell}}\}, \forall i \in \mathcal{A},\ \ell \in \mathcal{L}_i$.

et des contraintes:

1. $\forall (i,l) \in \mathcal{A} \times \mathcal{L}_i :\ e_{i_{preem}^{\ell}} = s_{i_{preem}^{\ell+1}}$
2. $\forall i \in \mathcal{A} :\ s_{i_{min}} = s_{i_{preem}^1}$ et $e_{i_{preem}^{|\mathcal{L}_i|}} = e_{i_{min}}$
3. $\forall t \in \mathcal{T}$:
$$\sum_{\substack{i \in \mathcal{A} \\ t \in [s_{i_{min}}, e_{i_{min}}[}} b_{i_{min}} + \sum_{\substack{(i,l) \in \mathcal{A} \times \mathcal{L}_i \\ t \in [s_{i_{preem}^{\ell}}, e_{i_{preem}^{\ell}}[}} b_{i_{preem}^{\ell}} \leq B$$

4. $\forall i \in \mathcal{A} : \sum_{l \in \mathcal{L}_i} \left(f_i(b_{i^\ell_{preem}})(e_{i^\ell_{preem}} - s_{i^\ell_{preem}}) \right) + f_i(b_{i_{min}})(e_{i_{min}} - s_{i_{min}}) \geq W_i$

La première contrainte permet d'ordonner les sous-tâches de i_{preem}. Ceci dans le but de faciliter la modélisation des autres contraintes. La seconde contrainte modélise le fait que i_{preem} commence et finit en même temps que i_{min}. La troisième contrainte assure que la capacité de la ressource n'est pas excédée en sommant, à un instant t, les consommations minimales des tâches en cours ainsi que les consommations des sous-tâches préemptives en cours. Enfin, la quatrième contrainte permet de s'assurer que chaque tâche reçoit au moins l'énergie requise W_i.

Un des avantages de cette formulation est qu'elle permet l'utilisation des algorithmes de propagation mis en places pour la contrainte cumulative tels que le time-table classique [1], disjonctif [3] ou associé au edge-finding [5], le raisonnement disjonctif [1], ou encore le raisonnement énergétique [2].

Cependant, certains de ces raisonnements peuvent être adaptés pour prendre en compte l'ensemble du problème. C'est le cas, par exemple, du raisonnement énergétique détaillé ci-dessous.

2.1 Raisonnement énergétique

Ce paragraphe présente un algorithme de propagation pour le DECSP basé sur le raisonnement énergétique défini pour le CECSP [4]. L'adaptation de ce rasionnnement au cas discret est quasi-directe. Cependant, nous rappelons les bases de celui-ci car nous l'utiliserons dans la suite pour déduire des inégalités valides pour le PLNE.

Le principe du raisonnement énergétique est de comparer la quantité de ressource disponible dans un intervalle avec la quantité minimale de ressource consommée par toutes les tâches dans cet intervalle.

Les configurations pour lesquelles la quantité de ressource requise par une tâche i dans l'intervalle $[t_1, t_2]$ est minimale correspondent toujours à une configuration où la tâche reçoit le maximum d'énergie possible, i.e. est ordonnancer à b_i^{max}, en dehors de $[t_1, t_2]$, tout en respectant les contraintes (1)–(5). Ceci correspond donc à une des configurations suivantes:

- la tâche est calée à gauche: ordonnancer à b_i^{max} durant $[\underline{s_i}, t_1]$;
- la tâche est calée à droite: ordonnancer à b_i^{max} durant $[t_2, \overline{e_i}]$;
- la tâche est centrée: ordonnancer à b_i^{max} durant $[\underline{s_i}, t_1] \cup [t_2, \overline{e_i}]$ ou ordonnancer à b_i^{min} durant $[t_1, t_2]$.

En effet, dans le dernier cas, il peut arriver qu'ordonnancer la tâche à b_i^{max} dans $[\underline{s_i}, t_1] \cup [t_2, \overline{e_i}]$ implique que la quantité d'énergie restant à apporter à la tâche dans $[t_1, t_2]$ ne soit pas suffisante pour ordonnancer la tâche à b_i^{min} durant $[t_1, t_2]$. Or, ceci impliquerait une violation de la contrainte (2). Dans ce cas, la tâche est donc ordonnancer à b_i^{min} durant l'intervalle $[t_1, t_2]$.

Alors la quantité de ressource requise par la tâche i dans $[t_1, t_2]$ est la quantité minimale requise par ces configurations.

Les intervalles $[t_1, t_2]$ sur lesquels appliquer ce test pour le CECSP sont décrits dans [4]. Pour le DECSP, nous devons considérer les projections de ces intervalles sur les entiers, i.e. $[a, b] \rightarrow [\lfloor a \rfloor, \lceil b \rceil]$. Les ajustements pour le CECSP s'adaptent aussi naturellement au DECSP à l'aide de cette même projection.

3 Modèle de programmation linéaire en nombres entiers

3.1 Modèle

La formulation proposée dans cet article est une formulation indexée par le temps. Elle est adaptée de la formulation décrite dans [4]. Dans ces formulations, l'horizon de temps est divisé en intervalle de taille 1 et est défini par: $\mathcal{T}_\mathcal{D}$. Notons que, par translation, nous pouvons toujours supposer que $\min_{i \in \mathcal{A}} \underline{s}_i = 0$.

Pour chaque activité $i \in \mathcal{A}$ et pour chaque instant $t \in \mathcal{T}_\mathcal{D}$, nous définissons deux variables binaires x_{it} et y_{it} pour modéliser le début et la fin des activités. La variable x_{it} (resp. y_{it}) prendra la valeur 1 si et seulement si l'activité i commence (finit) à l'instant t. Pour modéliser la consommation de ressource et l'apport en énergie, nous introduisons deux variables, b_{it} et w_{it} qui représentent respectivement la quantité de ressource consommée par l'activité i dans la période de temps t et l'énergie reçue par cette même activité durant cette période.

Par manque de place, ce modèle n'est pas entièrement décrit ici mais nous décrivons les contraintes permettant de lier les variables b_{it} et w_{it}, i.e. permettant de calculer l'énergie apportée à i dans la période t, w_{it} , en fonction de la consommation de ressource b_{it}. Nous donnons aussi le nombre de variables et de contraintes du modèle.

Les contraintes liant b_{it} et w_{it}, $\quad \forall t \in \mathcal{T}_\mathcal{D},\ \forall i \in \mathcal{A}$ sont les suivantes:

$$w_{it} = a_i b_{it} + c_i \left(\sum_{\tau=\underline{s}_i}^{t} x_{i\tau} - \sum_{\tau=\underline{s}_i+1}^{t} y_{i\tau} \right) \tag{6}$$

Cette contrainte nous permet de modéliser la fonction de rendement f_i, $\forall i \in \mathcal{A}$. En effet, $\left(\sum_{\tau=\underline{s}_i}^{t} x_{i\tau} - \sum_{\tau=\underline{s}_i+1}^{t} y_{i\tau} \right)$ est égale à 1 si et seulement si l'activité i est en cours à l'instant t. Dans ce cas là, la valeur de l'énergie apportée à i est bien $w_{it} = a_i b_{it} + c_i$. Le second cas se produit quand l'activité i n'est pas en cours à t. Dans ce cas, $b_{it} = 0$ implique $w_{it} = 0$.

Le modèle possède donc $2n|\mathcal{T}_\mathcal{D}|$ variables binaires, $2n|\mathcal{T}_\mathcal{D}|$ variables continues et au plus $3n + |\mathcal{T}_\mathcal{D}| * (6n + 1)$ contraintes.

Dans la suite de cette section, nous utilisons le raisonnement énergétique pour déduire des inégalités valides pour ce modèle.

4 Inégalités valides basées sur le raisonnement énergétique

Ce paragraphe décrit des inégalités valides déduites du raisonnement énergétique. Soit $\mathcal{R}$ l'ensemble des intervalles d'intérêt pour le raisonnement énergétique.

$$(x_{i\underline{s}_i} + y_{i\overline{e}_i} - 1)\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2-t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{7}$$

$$(x_{i\underline{s}_i} + \sum_{t=t_1}^{t_2} y_{it} - 1)\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2-t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{8}$$

$$(\sum_{t=t_1}^{t_2} x_{it} + y_{i\overline{e}_i} - 1)\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2-t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{9}$$

$$(1 - \sum_{t<t_1} x_{it} - \sum_{t>t_2} y_{it})\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2-t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{10}$$

$$(\sum_{t\leq t_1} x_{it} + \sum_{t\geq t_2} y_{it} - 1)\,\underline{b}(i,t_1,t_2) \leq B(t_2-t_1)$$
$$\forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{11}$$

L'inégalité (7) correspond au cas où la tâche est centrée et est ordonnancer à b_i^{max} durant $[\underline{s_i}, t_1] \cup [t_2, \overline{e_i}]$. En effet, cette inégalité n'est active que dans le cas où $(x_{i\underline{s}_i} + y_{i\overline{e}_i} - 1) = 1 \Rightarrow \left[x_{i\underline{s}_i} = 1 \wedge y_{i\overline{e}_i} = 1\right]$. Or, ceci implique la tâche commence à $\underline{s_i}$ et finit $\overline{e_i}$. Donc, la ressource disponible dans $[t_1, t_2[$ doit être suffisante pour donner la quantité de ressource minimale requise par i dans $[t_1, t_2[$ dans cette configuration. Dans tous les autres cas, l'inégalité devient $\sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq B(t_2-t_1)$ ou $\sum_{j\neq i}\underline{b}(j,t_1,t_2) - \underline{b}(i,t_1,t_2) \leq B(t_2-t_1)$.

Les inégalités (8), (9), (10), (11) correspondent respectivement au cas où i est calée à gauche, i est calée à droite, i est complètement incluse dans $[t_1, t_2]$, i est exécutée à b_i^{min} durant l'intervalle $[t_1, t_2]$ et sont déduites de la même façon que (7).

Ces inégalités seront ajoutées au modèle indexé par le temps décrit à la section 3 pour renforcer ce dernier.

5 Résultats Expérimentaux

Nous avons testé les différentes méthodes de résolution proposées dans cet article sur les instances de [4]. Les expérimentations ont été conduites sous le système

		1^{re} sol.		fin algo.	
	#tâches	temps(s)	écart	temps	%opt.
DEF	20	5.37	7.85	75.4	0.25
ER	20	8.4	10.6	78.9	0.22
DEF	25	4.6	4.4	83.8	0.17
ER	25	0.06	3.86	60.1	0.4
DEF	30	0.99	7.18	75.19	0.25
ER	30	5.66	7.53	75.8	0.25

Table 1: Résultats du PLNE avec et sans inégalités valides: ER et DEF resp. (TL 1000s)

	1^{re} sol.		fin algo.		
#tasks	time	deviation	time lim.	MIP dev.	%solved
10	0.03	8.7	1	0.13	100
20	0.06	9.3	5	0.26	100
25	0.07	10.2	10	0.11	100
30	0.07	10.6	20	3.9	100

Table 2: Résultats du modèle PPC

d'exploitation Ubuntu 64-bit 12.04 et les résultats sont calculés au moyen d'un processeur 4-core, 8 thread Core (TM) i7-4770 CPU et de 8GB de mémoire RAM.

Le modèle de PLNE est résolu à l'aide de IBM Cplex 12.6 avec 2 threads et une limite de temps de 100 secondes. Les inégalités déduites du raisonnement énergétique sont calculées avant la résolution du PLNE et ajoutées statiquement au modèle. Ceci augmente la taille du modèle de $5|\mathcal{R}|n$ contraintes (avec $|\mathcal{R}| \in O(n^2)$).

Le tableau 1 décrit les résultats du PLNE. Le gap de la première solution trouvée par les modèles (3^{me} colonne) est calculé à partir de la meilleure solution trouvée.

L'ajout des inégalités du raisonnement énergétique permet de résoudre les instances à 25 tâches de manières plus efficace. Cependant, elles ralentissent le modèle pour les instances à 20 ou 30 tâches mais la perte de rapidité dans ce cas là est beaucoup moins élevée que le gain fait sur les instances à 25. Une poursuite de recherche intéressante serait d'essayer d'ajouter ces contraintes pendant la résolution du PLNE en tant que coupes.

Le modèle de PPC est résolu avec IBM CP Optimizer 12.6. Le tableau 2 décrit les résultats du modèle de PPC. Le modèle de PPC est testé sans ajout du raisonnement énergétique présentés dans cet article mais des résultats expérimentaux plus détaillées seront proposés lors de la conférence.

Les résultats montrent l'intérêt des inégalités valides ajoutées au PLNE. Le modèle de PPC ne permet pas de prouver l'optimalité des solutions trouvées mais a des résultats similaires au PLNE.

Les méthodes présentées ont aussi été testées sur des instances mises à l'échelle de la manière suivante. Soit α le plus petit commun multiple à tous les b_i^{min} et b_i^{max}. Alors, la mise à l'échelle consiste à multiplier $\overline{e}_i$, $\underline{e}_i$, $\overline{s}_i$, $\underline{s}_i$ et

W_i par α. Ces expérimentations n'ont pas donné de résultats dû à la grande taille de ces modèles. Les modèles continus restent donc la seule alternative pour obtenir des solutions optimales dans le cas où la solution est réelle.

Parmi les poursuites de recherche possibles, on trouve l'amélioration des modèles avec la réduction du nombre de variable et/ou de contraintes et la mise en place d'algorithmes de propagation dédiés.

References

[1] P. Baptiste, C. Le Pape and W. Nuijten (2001). *Constraint-based scheduling*, Kluwer Academic Publishers, Boston/Dordrecht/London.

[2] J. Erschler and P. Lopez (1990). Energy-based approach for task scheduling under time and resources constraints. *2nd International Workshop on Project Management and Scheduling*, pp. 115-121, Compiègne, France.

[3] S. Gay, R. Hartert and P.Schaus (2015). Time-Table Disjunctive Reasoning for the Cumulative Constraint. *CPAIOR 2015, Proceedings*, pp 157–172.

[4] M. Nattaf, C. Artigues, P. Lopez and D. Rivreau (2015). Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions. *OR Spectrum*, pp 1–34.

[5] P. Vilím (2011). Timetable Edge Finding Filtering Algorithm for Discrete Cumulative Resources. *CPAIOR 2011, Proceedings*, pp 230–245.