Tâches couplées identiques : le cas fini

Michaël Gabay, Gerd Finke, Nadia Brauner

Tâches couplées identiques : le cas fini

Michaël Gabay[1], Gerd Finke[1], Nadia Brauner[1]

Laboratoire G-SCOP
46, avenue Félix Viallet - 38031 Grenoble Cedex 1 (France)
{michael.gabay,gerd.finke,nadia.brauner}@g-scop.grenoble-inp.fr

Mots-clés : *Ordonnancement, Tâches couplées, High multiplicity.*

1 Introduction

Un système radar traditionnel est composé d'une seule antenne permettant d'émettre et de recevoir des ondes radio. Après l'émission d'une onde, celle-ci doit voyager, être réfléchie, puis détectée et interprétée par le radar. Durant la période de voyage d'une onde, d'autres ondes peuvent être émises ou reçues.

En 1980, Shapiro [1] introduisit les tâches couplées pour modéliser ces opérations. Une tâche couplée i est une tâche composée de deux opérations : la première, de durée a_i (l'émission) et la seconde, de durée b_i (la réception). Ces deux opérations sont séparées par une durée fixe L_i correspondant à la durée de voyage de l'onde. Une tâche couplée est représentée Figure 1. L'objectif du problème est d'ordonnancer ces tâches sur une seule machine en minimisant la durée totale de l'ordonnancement. Shapiro [1] a prouvé que ce problème est $\mathcal{NP}-$complet.

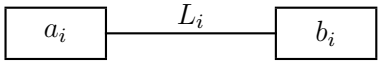

FIG. 1 – Une tâche couplée

Orman et Potts [2] ont étudié ce problème et décrit sa complexité pour différents cas particuliers (voir Tableau 1). Toutefois, le cas des tâches couplées identiques où $\forall i$, $a_i = a$, $b_i = b$, $L_i = L$ est resté ouvert. Ce cas particulier diffère des autres cas puisque la donnée du problème est alors composée de 4 entiers a, b, L et n. On peut supposer sans perte de généralité que $a > b$ (par symétrie du problème) et $a + b \leq L$ (sinon la solution est triviale). La taille de l'entrée est alors $O(log(L) + log(n))$ et un algorithme polynomial en n serait exponentiel en la taille de l'instance. Un ordonnancement n'est donc pas un certificat polynomial pour ce problème. Un tel problème est appelé problème d'ordonnancement en *high multiplicity*. Pour plus de détails sur ces problèmes, on peut se référer à [3, 4, 5, 6] par exemple.

Fortement $\mathcal{NP}$-Complet	$a_j = L_j = b_j$
	$a_j = b_j = p\,;\, L_j$
	$a_j = a\,;\, L_j = L\,;\, b_j$
Ouvert	$a_j = a\,;\, L_j = L\,;\, b_j = b$
Polynomial	$a_j = L_j = p\,;\, b_j$
	$a_j = b_j = p\,;\, L_j = L$

TAB. 1 – Complexité des cas particuliers du problème d'ordonnancement de tâches couplées

Ahr *et al.* [7] ont proposé un algorithme permettant de résoudre ce problème de manière exacte. Sa complexité est $O(nr^{2L})$ où $r \leq \sqrt[a-1]{a}$. Baptiste [8] a amélioré ce résultat et prouvé que pour a, b et L fixés, ce problème peut être résolu en un temps $O(log(n))$. Toutefois, la constante est exponentielle en L et la complexité de cet algorithme est donc exponentielle.

Lehoux-Lebacque *et al.* [9] ont résolu le cas cyclique. Dans ce cas, l'objectif est de maximiser le taux de production.

Nous avons travaillé sur le cas fini (n tâches) et montré que ce cas est notablement différent du cas cyclique. En particulier, aucune des propriétés cruciales permettant de résoudre le cas cyclique ne peut être appliquée au cas fini. Les preuves et détails des résultats énoncés dans ce résumé sont disponibles sur *HAL* [10].

2 Le cas cyclique

Lehoux-Lebacque *et al.* [9] ont proposé un algorithme polynomial (complexité $O(log(L)^2)$) pour le cas cyclique. Ils utilisent des patterns pour décrire les cycles et retournent un certificat concis. La preuve de leurs résultats repose sur certaines propriétés du cas cyclique dont :

- On place toujours le maximum d'opérations dans chaque temps mort L_i ;
- L'ordre des opérations dans un cycle est sans importance ;
- Le pattern des opérations entre les deux opérations d'une même tâche couplée est un invariant.

Ces propriétés permettent de créer un certificat polynomial en la taille de l'instance. Ce certificat est composé de 3 entiers et est de longueur $O(log(L))$.

Un exemple de cycle est exposé Figure 2.

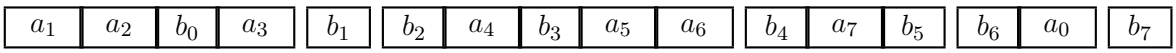

FIG. 2 – $a = 5$, $b = 4$, $L = 15$, un cycle optimal

3 Le cas fini

3.1 Du cas cyclique au cas fini

Partant du cas cyclique, nous espérions pouvoir généraliser ses solutions au cas fini. Éventuellement, en prolongeant cet ordonnancement au début et à la fin. Nous définissons une *stratégie pure* utilisant le cycle optimal comme la solution dans laquelle on répète ce cycle (en omettant les premiers b car leurs a n'ont pas été ordonnancés) jusqu'à avoir ordonnancé n opérations a puis on termine les dernières tâches commencées. Nous avons prouvé le théorème 1.

Théorème 1 *La stratégie pure utilisant le cycle optimal est asymptotiquement optimal.*

En fait, nous démontrons que cette stratégie est une $(1 + \frac{2}{k})$-approximation où k est le nombre de fois où le cycle est répété. Cet ordonnancement peut être décrit de manière polynomiale en la taille de l'instance et sa durée totale peut également être calculée en temps polynomial.

Ce résultat pose naturellement la question suivante : *La solution de la stratégie pure utilisant le cycle optimal est-elle optimale pour un n suffisamment grand ?*
Nous avons montré que ce n'est pas le cas. Plus précisément, il existe des instances telles que quelque soit la valeur de n, il existe un n' plus grand et tel que la durée totale d'une solution optimale soit strictement inférieure à la durée totale de la solution de la stratégie pure utilisant le cycle optimal.

3.2 Problèmes soulevés

Nous avons implémenté le problème comme un programme linéaire en nombres entiers et résolu de nombreuses instances en utilisant ce modèle et le solver commercial `CPLEX`. Ces tests nous ont permis de trouver de nombreux contre-exemples pour toutes les propriétés cruciales du cas cycliques. De plus, la forme de certaines solutions optimales est particulièrement complexe (voir Figure 3 par exemple) et deux instances différentes du problème peuvent avoir

des solutions très différentes. En particulier, en faisant simplement varier n pour des mêmes valeurs de a, b et L, les formes des solutions optimales peuvent varier considérablement. Ces différences marquées ne nous ont pas permis d'identifier de pattern (qui auraient peut-être permis d'obtenir un certificat polynomial) pour les solutions du cas fini. Concrètement, aucune des propriétés énoncées section 2 ne reste valide. En particulier, il manque, en quelque sorte, une récursivité entre les solutions optimales (pour a, b et L fixés, lorsqu'on augmente n).

Un exemple est illustré Figures 4 et 5. Sur cet exemple, la meilleure extension de l'ordonnancement Figure 4 donne un ordonnancement de durée totale 639. Pourtant une solution optimale est de durée totale 638 comme illustré Figure 5.

Enfin, nous avons tout de même pu montrer qu'une version affaiblie de certaines propriétés du cas cyclique peut rester vraie dans le cas fini sur certaines classes d'instances. Par exemple, pour certaines classes d'instances, on place toujours le nombre maximum d'opérations durant la durée séparant les deux opérations de la première tâche (en comptant éventuellement certains temps morts comme des b fantômes).

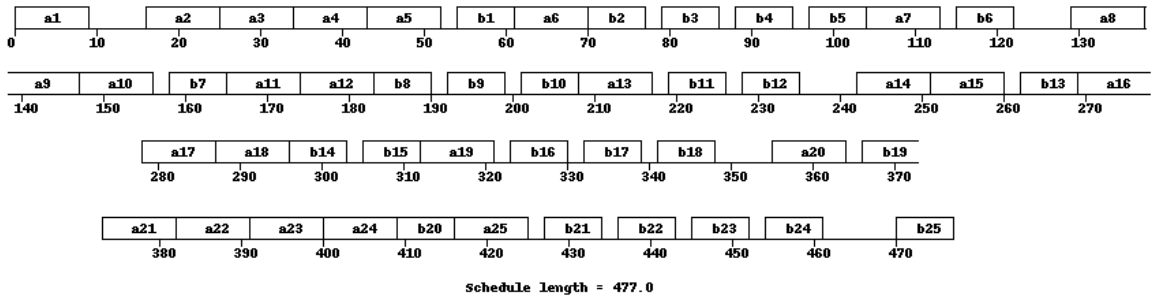

FIG. 3 – $a = 9$, $b = 7$, $L = 45$, $n = 25$; une solution optimale, durée 477

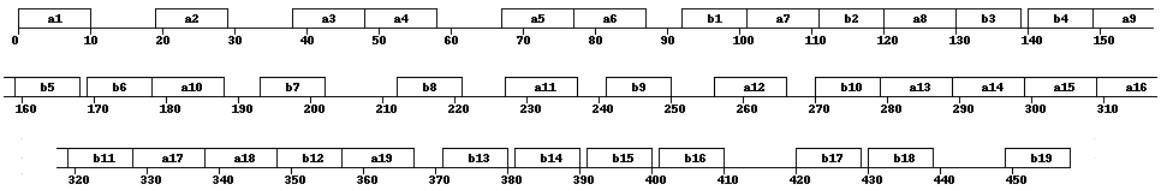

FIG. 4 – $a = 10$, $b = 9$, $L = 82$, $n = 19$; une solution optimale, durée 458

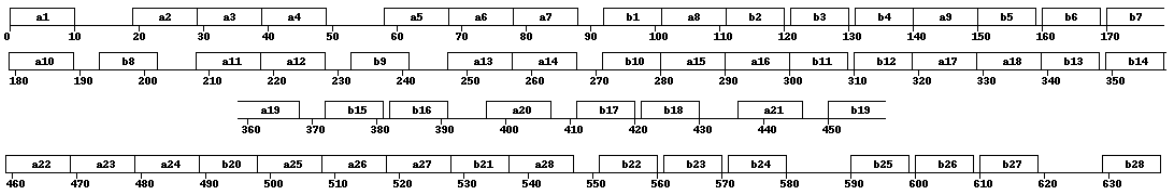

FIG. 5 – $a = 10$, $b = 9$, $L = 82$, $n = 28$; une solution optimale, durée 638

3.3 Un cas polynomial

Considérons à présent certains cas particuliers de ce problème. Par exemple, lorsqu'on a $\frac{L}{a} + 1 \geq n$, l'ordonnancement glouton (placer chaque opération dès que possible) est

clairement optimal. Nous avons également trouvé les solutions optimales pour certaines valeurs de n, lorsque $a = b + 1$ et que le cycle optimal est d'une certaine forme. Le résultat peut être obtenu et vérifié en temps polynomial. Dans ces cas, les solutions optimales sont non triviales et requièrent d'utiliser différents pattern, connectés par une transition, comme représenté Figure 4.

Ces cas sont résolus mais ils restent très spécifiques. De plus, les solutions sont bien plus compliquées qu'on aurait pu l'imaginer pour un cas aussi simple. Tous les autres cas restent ouverts et avoir un algorithme pour chaque valeur de n et chaque type de cycle optimal semble difficile.

4 Conclusion

Nous avons montré que le cas fini est très différent du cas cyclique, essentiellement car les principaux résultats et propriétés du cas cyclique ne s'appliquent pas et ne peuvent pas être directement étendus. Nous avons résolu un cas particulier du problème pour lequel nous proposons une solution optimale calculable et vérifiable en temps polynomial. Toutefois, la spécificité et la difficulté de ce simple cas ne sont pas encourageants pour la suite et renforcent l'idée qu'il est peut être impossible de trouver un algorithme polynomial pour résoudre ce problème, voire même impossible d'avoir un certificat polynomial. Le problème reste ouvert mais nos résultats laissent à penser qu'il n'est probablement pas dans $\mathcal{NP}$. Cette conjecture doit encore être prouvée, par exemple en réduisant un problème EXPSPACE-complet à celui-ci.

Références

[1] R.D. Shapiro. Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27(3) :489–498, 1980.

[2] A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72(1-2) :141–154, 1997.

[3] D.S. Hochbaum and R. Shamir. Minimizing the number of tardy job units under release time constraints. *Discrete Applied Mathematics*, 28(1) :45–57, 1990.

[4] D.S. Hochbaum and R. Shamir. Strongly polynomial algorithms for the high multiplicity scheduling problem. *Operations Research*, 39(4) :648–653, 1991.

[5] J.J. Clifford and M.E. Posner. Parallel machine scheduling with high multiplicity. *Mathematical programming*, 89(3) :359–383, 2001.

[6] N. Brauner, Y. Crama, A. Grigoriev, and J. van de Klundert. A framework for the complexity of high-multiplicity scheduling problems. *Journal of combinatorial optimization*, 9(3) :313–323, 2005.

[7] D. Ahr, J. Békési, G. Galambos, M. Oswald, and G. Reinelt. An exact algorithm for scheduling identical coupled tasks. *Mathematical Methods of Operations Research*, 59(2) :193–203, 2004.

[8] P. Baptiste. A note on scheduling identical coupled tasks in logarithmic time. *Discrete Applied Mathematics*, 158(5) :583–587, 2010.

[9] V. Lehoux-Lebacque, N. Brauner, and G. Finke. Identical coupled task scheduling : polynomial complexity of the cyclic case. *Cahiers Leibniz 179*, 2009.

[10] Michaël Gabay, Gerd Finke, and Nadia Brauner. Identical coupled task scheduling problem : the finite case. *HAL, http ://hal.archives-ouvertes.fr/hal-00627902*. Technical report, 2011.