Une approche de programmation dynamique pour le problème de séquencement des tâches et de changement d'outil

Emma Legrand[1], Vianney Coppé[2], Daniele Catanzaro[3], Pierre Schaus[1]

[1] ICTEAM, UCLouvain, Belgique
[2] Hexaly, France
[3] CORE, UCLouvain, Belgique

25 février 2025

Résumé

Nous présentons un algorithme exact basé sur la programmation dynamique pour le problème de séquencement des tâches et de changement d'outil, un problème combinatoire issu des systèmes de fabrication. Nous introduisons une nouvelle famille de bornes inférieures plus serrées que celles existantes, améliorant la qualité des solutions et l'efficacité de l'élagage. Nous utilisons A et ses variantes anytime pour explorer l'espace des solutions, ainsi qu'une structure de données appelée FreeTools pour suivre l'état et calculer efficacement les coûts incrémentaux. Nos expériences montrent que notre approche surpasse les méthodes de l'état de l'art, notamment la programmation linéaire en nombres entiers.*

Mots-clés

Optimisation combinatoire, séquencement des tâches, changement d'outil, séparation et évaluation, A, programmation dynamique.*

Abstract

We present an exact algorithm based on dynamic programming for the Job Sequencing and Tool Switching Problem (JS-TSP), a combinatorial problem derived from manufacturing systems. We introduce a new family of lower bounds that are tighter than existing ones, improving solution quality and pruning efficiency. We use A and its anytime variants to explore the solution space and a data structure called FreeTools to track state and compute incremental costs efficiently. Our experiments show that our approach outperforms state-of-the-art methods, notably integer linear programming.*

Keywords

Combinatorial optimization, job sequencing, tool switching, branch and bound, A, dynamic programming.*

1 Introduction

Le *problème de séquencement des tâches et de changement d'outil* est un problème d'optimisation combinatoire issu des systèmes de fabrication. Il consiste à ordonnancer une séquence de tâches sur une machine flexible $\mathcal{M}$ qui peut être configurée avec un ensemble de maximum c outils parmi un ensemble $T = 0, 1, 2, \ldots, m-1$ de $m \geq 3$ outils. Nous appelons c la *capacité* de $\mathcal{M}$. De plus, nous considérons un ensemble $J = 1, 2, \ldots, n$ de $n \geq 3$ tâches à exécuter séquentiellement sur $\mathcal{M}$. Chaque tâche $j \in J$ nécessite un ensemble d'outils $t(j) \subseteq T$, avec $|t(j)| \leq c$. Lorsque $\mathcal{M}$ passe d'une tâche à une autre, il est nécessaire de choisir quels outils conserver et lesquels remplacer, cette opération est appelée *changement d'outil*.

Ce problème consiste à trouver une séquence de tâches qui minimise le nombre total de changements d'outils nécessaires. Tang et Denardo ont démontré en 1987 qu'il est *NP-difficile* et ont proposé la première formulation en programmation linéaire en nombres entiers (ILP) [7]. D'autres approches ont été développées [4, 2, 1], notamment des méthodes *Branch-and-Bound* (B&B) [5]. Nous nous penchons uniquement sur les méthodes de résolution exactes. Un récent article [6] a été publié sur ce sujet. Cependant, il n'a pas été inclus dans la comparaison car il a été publié après la soumission de l'article.

Tâches	1	2	3	4	5	6
Outils	0	1	1	0	5	3
	1	3	5	4	6	5
	2	4	6	6	7	
			8	8		
Capacité : 4						

TABLE 1 – Exemple d'une instance avec 9 outils et 6 jobs.

Tâches	1	2	4	3	5	6
Outils	0	(0)	0	1	6	3
	1	1	6	6	7	5
	2	3	8	8	5	
		4	4	5		
Capacité : 4						

TABLE 2 – Exemple d'une solution optimale pour l'instance 1 pour laquelle la valeur optimale est égale à 11.

Par exemple, la Table 2 montre la séquence de travail qui minimise le nombre total de changements d'outils pour l'instance présentée dans la Table 1. La valeur optimale pour cette instance est égale à 11. Les outils soulignés sont ceux qui donnent lieu à un changement d'outil, tandis que les outils encerclés ont la particularité de ne pas être immédiatement nécessaires pour traiter un travail spécifique j dans la séquence, mais d'être nécessaires pour traiter les tâches qui suivent j. Cette propriété, introduite pour la première fois par [7], donne lieu à un algorithme gourmand, communément appelé *Keep Tool Needed Soonest* (KTNS), qui calcule en un temps polynomial les changements d'outils optimaux pour une séquence de travail fixe. Cependant, ces approches restent limitées pour des grandes instances.

2 Contributions et conclusion

Nos contributions principales incluent l'introduction d'une nouvelle famille de bornes inférieures, qui est plus serrée que les bornes existantes [3, 5]. Cette famille de bornes combine les avantages des bornes déjà présentes dans l'état de l'art en utilisant une approche basée sur les arbres couvrants minimaux (MST). Dans ces arbres, les noeuds représentent une tâche ou un ensemble de tâches et les arêtes représentent le coût minimum pour passer d'une tâche à une autre. Cette amélioration permet de réduire significativement l'espace de recherche en élaguant plus efficacement les noeuds non prometteurs. Nous proposons également une version incrémentale de l'algorithme KTNS, appelée Lazy-KTNS, qui calcule de manière incrémentale le coût des séquences partielles. Cet algorithme utilise une structure de données spécifique, appelée FreeTools, pour garder une trace des outils pouvant être réutilisés gratuitement dans les étapes ultérieures. Cette approche réduit la complexité temporelle de $\mathcal{O}(n^2 \cdot m)$ à $\mathcal{O}(n \cdot c^2)$.
La Figure 1 permet d'observer que pour la majorité des instances l'algorithme Lazy-KTNS prend moins de temps pour trouver et prouver l'optimalité que l'algorithme KTNS.

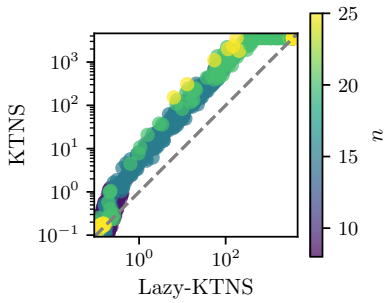

FIGURE 1 – Comparaison de la méthode B&B avec les algorithmes KTNS et Lazy-KTNS (en sec).

Enfin, nous reformulons le problème en termes de recherche en graphe, en utilisant l'algorithme A* et ses variantes anytime (comme Anytime Column Search et Iterative Beam Search). Cette reformulation permet d'éviter l'exploration redondante des séquences partielles équivalentes, ce qui réduit le nombre de nœuds explorés. Nous montrons que cette approche est plus efficace que les méthodes de B&B traditionnelles, en particulier pour les instances de grande taille et avec Anytime Column Search.
Nos expériences montrent que notre approche surpasse les méthodes existantes, y compris les formulations ILP, en termes de temps de calcul et de capacité à résoudre des instances de grande taille. Par exemple, notre méthode est capable de résoudre 100% des instances jusqu'à 15 jobs en moins d'une heure, alors que les formulations ILP existantes n'y arrivent pas.
En conclusion, notre travail propose une nouvelle approche pour résoudre le problème de séquencement des tâches et de changement d'outil, en combinant des bornes inférieures améliorées, un algorithme de calcul incrémental des coûts et une reformulation en termes de recherche en graphe. Ces contributions permettent de résoudre efficacement des instances de grande taille, ce qui était auparavant difficile avec les méthodes existantes. Pour des travaux futurs, nous envisageons d'explorer des variantes de la borne et d'étudier l'application de techniques de programmation par contraintes pour résoudre ce problème.

Références

[1] Christian Bessiere, Emmanuel Hebrard, Marc-André Ménard, Claude-Guy Quimper, and Toby Walsh. Buffered resource constraint : Algorithms and complexity. In *Integration of AI and OR Techniques in Constraint Programming : 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings 11*, pages 318–333. Springer, 2014.

[2] Daniele Catanzaro, Luis Gouveia, and Martine Labbé. Improved integer linear programming formulations for the job sequencing and tool switching problem. *European journal of operational research*, 244(3) :766–777, 2015.

[3] Yves Crama, Alwin G Oerlemans, Frits CR Spieksma, Yves Crama, Alwin G Oerlemans, and Frits CR Spieksma. *Minimizing the number of tool switches on a flexible machine*. Springer, 1996.

[4] Gianpaolo Ghiani, Antonio Grieco, and Emanuela Guerriero. Solving the job sequencing and tool switching problem as a nonlinear least cost hamiltonian cycle problem. *Networks : An International Journal*, 55(4) :379–385, 2010.

[5] Gilbert Laporte, Juan Jose Salazar-Gonzalez, and Frederic Semet. Exact algorithms for the job sequencing and tool switching problem. *IIE transactions*, 36(1) :37–45, 2004.

[6] Yuzhuo Qiu, Mikhail Cherniavskii, Boris Goldengorin, and Panos M. Pardalos. A computational study of the tool replacement problem. *INFORMS Journal on Computing*.

[7] C. S. Tang and E. V. Denardo. Models arising from a flexible manufacturing machine, Part I : Minimization of the number of tool switches. *Operations Research*, 36(5) :767–777, 1987.