Une nouvelle heuristique pour le problème d'ordonnancement à machines parallèles avec dates de disponibilité et temps de latence

A. Gharbi[1,2] et M. Haouari[2]

[1] Institut Supérieur d'Informatique, 2 Rue Abou Raihane Bayrouni, 2080 Ariana, Tunisie
anis.gharbi@ept.rnu.tn
[2] CORG-Laboratoire d'Ingénierie Mathématique, Ecole Polytechnique de Tunisie, BP 743, 2078 La Marsa, Tunisie
mohamed.haouari@ept.rnu.tn

1 Introduction

On considère le problème d'ordonnancement à machines parallèles décrit de la manière suivante : Un ensemble J de n tâches doit être ordonnancé sur m machines identiques ($n > m > 1$). A chaque tâche j ($j = 1, ..., n$) on associe un temps d'exécution p_j, une date r_j à partir de laquelle la tâche devient disponible, et un temps de latence q_j. Toutes les données sont déterministes et entières. Chaque machine exécute au plus une tâche à la fois et chaque tâche ne peut être exécutée que par une seule machine à la fois. On suppose que l'exécution d'une tâche ne peut pas être interrompue et que toutes les machines sont disponibles à partir de l'instant zéro. L'objectif est de déterminer un ordonnancement qui minimise le makespan (i.e. la date de fin de la dernière tâche ordonnancée). Ce problème, noté $P|r_j, q_j|C_{max}$, est NP-difficile au sens fort. Outre ses applications pratiques, le $P|r_j, q_j|C_{max}$ apparaît comme une bonne relaxation du problème de Flow Shop Hybride. Aussi, il joue un rôle central dans certains algorithmes de résolution exacte du problème d'ordonnancement de projet avec contraintes de ressources. Le $P|r_j, q_j|C_{\max}$ a fait l'objet de plusieurs travaux, la plupart étant sur des bornes inférieures ou des méthodes exactes. Cependant, très peu d'heuristiques ont été proposées dans la littérature pour ce problème. Carlier [1] et Gusfield [4] ont étudié l'algorithme de Jackson, qui est un simple algorithme de liste. Plus récemment, des méthodes basées sur des parcours tronqués d'arborescence ont été proposées par Tercinet [5]. Ces méthodes ont été testées sur des instances allant jusqu'à 100 tâches.

L'objectif de ce travail est double. En premier lieu, nous montrons que la déviation maximale de l'algorithme de Jackson par rapport à l'optimum est améliorée si celui-ci est précédé par un algorithme de prétraitement. Ensuite, nous proposons une nouvelle heuristique pour le $P|r_j, q_j|C_{\max}$. Cette dernière est basée sur la résolution, par un algorithme de branch-and-bound tronqué, d'une séquence de sous-problèmes à deux machines. Des expérimentations numériques réalisées sur un grand nombre d'instances montrent que l'approche proposée permet d'obtenir rapidement des solutions très proches de l'optimum et ceci pour des instances de très grande taille.

2 Une amélioration de l'algorithme de Jackson

L'algorithme de Jackson constitue une méthode simple et rapide pour obtenir un assez bon ordonnancement pour le $P|r_j, q_j|C_{\max}$. Cet algorithme est basé sur la règle de priorité qui consiste à ordonnancer sur la première machine disponible, la tâche en attente ayant le plus grand temps de latence. Carlier [1] et Gusfield [4] ont montré que la déviation absolue du makespan fourni par l'algorithme de Jackson par rapport à l'optimum est inférieure ou égale à $\min\{\lceil(2 - 1/m)\max_{j\in J} p_j\rceil - 1, 2(\max_{j\in J}\ p_j - 1)\}$.

Gharbi et Haouari [2] ont proposé un algorithme de prétraitement qui permet de fixer la date de début de certaines tâches de façon à réduire l'ensemble J des tâches à ordonnancer à un sous-ensemble $\bar{J}$. De plus, ils montrent qu'étant donné un ordonnancement de $\bar{J}$ de makespan C, un ordonnancement complet de J de makespan $\max\left\{C, \max_{j\in J\setminus\bar{J}}(r_j + p_j + q_j)\right\}$ peut être reconstitué en un temps $O(n \log n)$.

Nous montrons que l'application de l'algorithme de prétraitement suivie de celui de Jackson sur $\bar{J}$ puis la reconstitution d'un ordonnancement complet fournit un makespan dont la déviation

maximale par rapport à l'optimum est meilleure que celle de Jackson. En effet, cette déviation maximale est égale à $\min\{\lceil(2-1/m)\max_{j\in\bar{J}} p_j\rceil - 1, 2(\max_{j\in\bar{J}} p_j - 1)\}$.

3 Un algorithme de décomposition

Etant donné un ordonnancement σ, désignons par J_k $(k = 1, ..., m)$ le sous-ensemble de tâches ordonnancées sur la machine M_k. On pose $C_k = \max_{j\in J_k} C(j)$ où $C(j)$ désigne le temps de fin de la tâche j. Nous supposons que $C_1 \leq C_2 \leq ... \leq C_m$.

La procédure proposée consiste à sélectionner itérativement une paire de machines (M_k, M_m) $(1 \leq k \leq m-1)$ et résoudre l'instance à deux machines $P2|r_j, q_j|C_{\max}$ définie sur le sous-ensemble $J_k \cup J_m$. Si le makespan obtenu, noté $C^k_{\max}$, est plus petit que C_m, alors les ordonnancements des machines M_k et M_m dans σ, sont remplacés par les nouveaux. La procédure est réitérée jusqu'à avoir $C^k_{\max} = C_m \ \forall\ k = 1, ..., m-1$. On commence par résoudre le sous-problème défini par (M_1, M_m). Si $C^1_{\max} < C_m$, alors les valeurs de C_k sont mises à jour et la procédure est réitérée. Sinon, c'est le sous-problème défini par (M_2, M_m) qui est résolu, et ainsi de suite.

Afin d'obtenir rapidement une bonne solution, les instances à deux machines sont résolues à l'aide d'une version tronquée de l'algorithme de branch-and-bound décrit dans [3]. Plus précisément, cet algorithme est arrêté lorsque, soit un temps limite T_{max} est atteint, soit une solution à $\varepsilon\%$ de l'optimum est obtenue, soit un ordonnancement de makespan strictement inférieur à C_{m-1} est obtenu. Dans notre implémentation, nous avons fixé T_{max} à $\lceil\frac{n}{20m}\rceil$ et ε à 0.5. L'ordonnancement initial est fourni par l'algorithme de Jackson amélioré.

4 Résultats numériques préliminaires

Notre approche a été implémentée sur un Pentium IV 2.8 GHz, et a été testée sur une série d'instances où tous les r_j, p_j et q_j sont générés uniformément entre 1 et n. Le tableau 1 donne le gap relatif par rapport à une borne inférieure, ainsi que le temps moyen de calcul de notre heuristique. Nous observons que la méthode proposée fournit des solutions très proches de l'optimum (Gap souvent inférieur à 0.1%), et ce en un temps moyen très court pour les moyennes tailles (moins de 15 secondes), mais elle requiert jusqu'à environ 9 minutes pour les grandes tailles. Cependant, l'heuristique devient bien plus rapide si elle est stoppée dès qu'une solution à moins de 0.5% d'une borne inférieure est obtenue. En effet, nos expérimentations ont montré qu'elle prend en moyenne environ 2 minutes pour fournir de telles solutions pour des instances à 2000 tâches et 100 machines.

Il est à noter que l'heuristique proposée a été testée sur une deuxième série d'instances, générée de la même manière que dans [1][2][5]. Nous avons constaté que ces instances sont faciles à résoudre. En effet, notre heuristique a pu résoudre à l'optimalité 97.50% des instances, et même les solutions fournies par l'algorithme de Jackson amélioré étaient optimales ou très proches de l'optimum.

TAB. 1. Performance de l'heuristique

	Instances de moyenne taille									Instances de grande taille								
n	100			300			500			1000			1500			2000		
m	5	10	20	5	10	20	5	10	20	40	80	100	40	80	100	40	80	100
Gap	0.08	0.26	1.13	0.07	0.08	0.09	0.07	0.07	0.09	0.09	0.09	0.12	0.09	0.09	0.11	0.09	0.09	0.09
Temps	1.27	2.17	1.14	5.64	4.03	14.71	15.61	11.92	10.86	73.23	335.40	536.60	171.27	240.80	405.01	280.40	265.91	365.37

Références

1. Carlier, J. : Scheduling jobs with release dates and tails on identical machines to minimize the makespan. European Journal of Operational Research (1987) ; 29, 298-306.
2. Gharbi A., Haouari M. : Minimizing Makespan on Parallel Machines Subject to Release Dates and Delivery Times. Journal of Scheduling (2002) ; 5, 329-355.
3. Gharbi A., Haouari M. : Optimal parallel machines scheduling with availability constraints. Discrete Applied Mathematics (à paraître).
4. Gusfield D. : Bounds for naive multiple machine scheduling with release times and deadlines. Journal of Algorithms (1984) ; 5, 1-6.
5. Tercinet F. : Méthodes arborescentes pour la résolution des problèmes d'ordonnancement, conception d'un outil d'aide au développement. Thèse de doctorat, Université François Rabelais Tours (2004).