Méthodes de résolution pour la caractérisation des solutions optimales du problème $1||L_{max}$

Tifenn Rault[1], Ronan Bocquillon[1], Jean-Charles Billaut[1]

Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
64 Avenue Jean Portalis, 37200 Tours
{tifenn.rault,ronan.bocquillon,jean-charles.billaut}@univ-tours.fr

Mots-clés : *ordonnancement, caractérisation de solutions, treillis de permutations.*

1 Introduction

Beaucoup de problèmes d'ordonnancement peuvent être résolus en temps polynomial grâce à une règle de priorité (SPT pour $1||\sum C_j$ et EDD pour $1||L_{max}$ par exemple). Cependant, ces problèmes peuvent admettre un grand nombre – possiblement exponentiel – de solutions optimales. L'objectif de notre étude est de caractériser l'ensemble des solutions optimales pour le problème $1||L_{max}$, sans les énumérer. Une telle approche se révèle intéressante dans le cadre de l'ordonnancement multicritère, ou dans un contexte dynamique pour assurer une certaine forme de robustesse.

Dans ce papier, nous nous intéressons au problème $1|\widetilde{d}_j|\sum N_j$ étudié par Ta et Billaut [3], qui consiste à chercher une solution faisable de plus bas niveau dans un treillis de permutations. Cette séquence permet à elle seule de caractériser un très grand nombre de séquences réalisables. Nous avons amélioré le Branch-and-Bound existant grâce à deux nouvelles règles de dominance, et nous avons proposé une nouvelle heuristique qui offre un bon compromis entre la qualité des solutions obtenues et le temps de résolution.

2 Description du problème

Nous considérons un ensemble de n tâches $J_j, 1 \leq j \leq n$. A chaque job sont associés une date due d_j et un temps de traitement p_j. Les pré-traitements suivants sont réalisés en temps polynomial : 1) L^*_{max} est obtenu en appliquant EDD, 2) les dates dues deviennent des deadlines $\widetilde{d}_j = \min(d_j + L^*_{max}, \sum p_j), \forall j \in \{1, 2, .., n\}$, 3) les tâches sont renumérotées selon EDD, et en cas d'égalité selon LPT. A la suite des pré-traitements, une solution faisable, c'est-à-dire satisfaisant les deadlines, est aussi une solution optimale pour le problème $1||L_{max}$. De plus, on sait que la séquence $\sigma = (J_1, J_2, ..., J_n)$ est faisable.

Le treillis de permutations est un graphe direct dont la racine est la séquence EDD $= (J_1, J_2, ..., J_n)$. Les enfants d'une séquence sont créés en permutant deux tâches J_k et J_l ssi J_k est juste avant J_l et $k < l$. Le processus est répété pour les nouveaux nœuds crées jusqu'à ce que le puits (la séquence inverse EDD) soit atteint. Une propriété intéressante du treillis de permutations est que les prédécesseurs d'une solution faisable sont aussi faisables, et ces solutions peuvent être caractérisées facilement en déduisant un ordre partiel entre les tâches. Chaque séquence dans le treillis peut être associée à un *niveau*, qui représente le nombre d'inversions par rapport à la séquence inverse EDD (le nombre de fois que l'on a J_i avant J_j et $i < j$). Ce niveau est également le nombre de contraintes d'ordre partiel qui caractérise cette séquence. Plus de détails sur le treillis de permutations peuvent être trouvés dans [1].

Notre objectif est de trouver la séquence faisable de plus petit niveau. Nous espérons ainsi caractériser un grand nombre de solutions. Ce problème est noté $1|\widetilde{d}_j|\sum N_j$ et sa complexité reste ouverte.

3 Méthodes de résolution

Méthode exacte Le B&B original construit les solutions par la fin et considère deux ensembles : $S = \{J_n, J_{n-1}, ..., J_1\}$ l'ensemble des tâches non ordonnancées et $\sigma = \emptyset$ la séquence en construction. Tant que $S \neq \emptyset$, on choisit un job dans S, du plus petit indice au plus grand indice, et on l'ajoute devant σ, si les deadlines et les conditions de dominance sont respectées. Nous proposons deux nouvelles règles de dominance :

> **Propriété 1** : *Si deux tâches J_i et J_j ont des deadlines identiques et que $i < j$, alors la tâche J_i est ordonnancée après la tâche J_j.*
>
> **Propriété 2** : *Si deux solutions partielles contiennent exactement le même sous-ensemble de tâches, alors la solution partielle qui a le plus petit niveau domine l'autre.*

La propriété 2 est intéressante car nous savons par construction que, dans un tel cas, c'est toujours la meilleure de ces solutions partielles qui est visitée en premier. Cela nous permet de mettre en place une stratégie de mémorisation consistant à enregistrer les solutions partielles déjà visitées. De cette façon, si nous rencontrons plus tard un nœud dont la solution partielle σ contient le même sous-ensemble de tâches qu'un autre nœud déjà exploré, le nouveau nœud sera ignoré.

Méthode heuristique Nous avons proposé une nouvelle heuristique basée sur LDS [2]. Durant l'exploration, LDS-k autorise la génération de chemins contenant au plus k branches "allant à droite". Notons qu'à l'origine LDS est une méthode itérative, que nous utilisons ici comme une méthode arborescente tronquée. L'idée qui a guidé ce choix est de chercher à trouver une séquence faisable le plus à gauche possible de l'arbre. En effet, lorsque l'on déroule l'arbre de recherche, on peut observer qu'au niveau des feuilles, les solutions sont quasiment réparties de la gauche vers la droite, du plus petit niveau au plus grand niveau.

Résultats préliminaires Nous avons testé nos méthodes sur les instances de Ta et Billaut [3]. Les propriétés 1 et 2 nous ont permis de résoudre à l'optimal 9 nouvelles instances, et de réduire légèrement le temps de résolution pour des instances de taille supérieure à 70. Pour notre approche heuristique, LDS-2 permet d'obtenir des gaps à la meilleure solution connue inférieurs à 12%, ce qui est bien moins que l'heuristique de liste présentée dans [3] qui atteint des gaps de 45%. Cette amélioration a un léger coût en temps de calcul, mais qui reste inférieur à celui du B&B.

4 Conclusions et perspectives

Nous avons proposé deux nouvelles règles de dominance nous ayant permis de résoudre 9 nouvelles instances à l'optimal. Notre heuristique offre un bon compromis entre la qualité des solutions obtenues et le temps de résolution. En ce qui concerne les perspectives, nous souhaitons générer des instances plus difficiles, et nous souhaitons étudier différentes politiques de gestion de mémoire lorsque la base de données utilisée pour la mémorisation est pleine.

Références

[1] J.-C. Billaut, E. Hebrard and P. Lopez. *Complete Characterization of Near-Optimal Sequences for the Two-Machine Flow Shop Scheduling Problem.* Ninth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming, Nantes, France, 2012.

[2] W. D. Harvey and M. L. Ginsberg. *Limited Discrepancy Search.* Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 607-613, 1995.

[3] T.T.T. Ta. *New single machine scheduling problems with deadlines for the characterization of optimal solutions.* Thèse de doctorat, Université de Tours, 2018.