Heuristique à divergence limitée pour les problèmes d'ordonnancement avec contraintes de délais

Wafa Karoui[1,2,3], Marie-José Huguet[1,2], Pierre Lopez[1,2], Mohamed Haouari[3]

[1] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
[3] Unité de recherche ROI ; Ecole Polytechnique de Tunisie, 2078 La Marsa, Tunisie
{wakaroui,huguet,lopez}@laas.fr, mh6368@yahoo.com

Mots-Clés : *ordonnancement, time lags, heuristique, divergence.*

1 Problèmes d'ordonnancement avec time lags

Dans ce travail, nous nous intéressons aux problèmes d'ordonnancement de type flowshop et jobshop avec contraintes de délais plus connues sous le nom de *time lags*. Notre objectif est de minimiser la durée totale de l'ordonnancement (*makespan*) d'un problème considéré. Les contraintes de time lags peuvent être définies de plusieurs manières. Elles ont été présentées la première fois par Mitten en 1958 [3] comme étant le temps entre la fin d'une opération et le début d'une autre. Dans notre cas, il s'agit de contraintes entre les opérations successives d'un même job. Ainsi, la distance temporelle séparant ces opérations comporte une limite inférieure et une limite supérieure qu'on appelle respectivement contraintes de time lags minimum et maximum et qui viennent se rajouter aux contraintes initiales du problème considéré. Ces problèmes sont de difficulté au moins égale à celle des problèmes initiaux considérés sans time lags (NP-difficiles au sens fort). Peu de méthodes de résolution ont été utilisées à ce jour pour résoudre les problèmes de jobshop avec time lags. Citons les travaux de [1], en particulier une méthode tabou et un algorithme évolutionnaire mémétique [1]. Ce dernier se révèle efficace pour les jobshops avec time lags maximaux nuls (problèmes dits *no-wait*).

2 Divergence et apprentissage pour les problèmes avec time lags

La méthode de résolution que nous proposons pour résoudre les problèmes d'ordonnancement avec time lags est une variante de *Climbing Discrepancy Search* (CDS) [2], un principe de recherche à base de divergence pour l'optimisation. Partant d'une solution initiale, cette méthode tente de l'améliorer en augmentant les écarts à cette solution (divergences) autorisées d'une itération à une autre. Toute solution meilleure que la solution initiale est mémorisée et adoptée comme nouvelle solution initiale dans l'itération suivante. Pour la terminaison de la recherche, une condition d'arrêt est fixée au préalable, par exemple une limite sur le temps d'exécution. Pour le paramétrage de notre méthode à la nature des problèmes étudiés, nous avons étudié plusieurs possibilités sur le positionnement des divergences dans l'arbre de recherche, le choix de l'heuristique d'initialisation de la solution, ainsi que la prise en compte d'un apprentissage basé sur des poids associés aux jobs.

En associant à un job sa durée D égale à la somme des durées de ses opérations ainsi que la durée de ses time lags DTL, on peut considérer les solutions initiales obtenues par les heuristiques qui classent les jobs en suivant l'ordre lexicographique ou bien l'ordre croissant ou décroissant des D, DTL, $D + DTL$ ou D/DTL.

Pour adapter la méthode au problème ciblé, nous avons associé à tous les jobs des poids initialement identiques. L'incrémentation d'un poids w_i associé au job J_i peut être gérée de différentes manières. On peut incrémenter w_i à chaque fois que : 1) une opération de J_i n'a pu être insérée dans le premier créneau disponible ; 2) sur une machine donnée, une opération n'a pu être placée dans le premier créneau disponible ; 3) une ou plusieurs opérations de J_i n'ont pu être placées dans les premiers créneaux disponibles sur les machines qui leur sont associées. Quel que soit le choix d'incrémentation, le poids d'un job donné est mis à jour à la fin de chaque itération en rajoutant à son ancienne valeur une nouvelle valeur obtenue en calculant la somme des poids que ce job a obtenu dans l'itération, ou bien en considérant le maximum de ces poids. Dans les itérations suivantes, ces poids sont considérés en plus de l'heuristique de base pour choisir le job à ordonnancer en priorité.

3 Évaluation des différentes propositions

Nous avons considéré toutes les combinaisons possibles des propositions avancées sur des instances classiques de problèmes d'ordonnancement proposées dans la *OR-library*. Pour les jobshops, nous considérons les instances $\{laX\}_{X=1..20}$ de Lawrence ainsi que les instances $ft06$ et $ft10$ de Fisher et Thompson. Pour les flowshops, nous considérons les instances $\{carY\}_{Y=5..8}$ de Carlier. Aux instances de base viennent se rajouter des instances avec contraintes de time lags maximaux générées à partir de cœfficients max égal à 0, 0.25, 0.5, 1, 2, 3, 5 et 10. Le délai considéré étant de 200 secondes, nous nous sommes comparé aux résultats obtenus par ILOG-Scheduler et par [1] (considère uniquement les $ft06$ avec max de 0, 0.5, 1 et 2, $\{laX\}_{X=1..20}$ avec max de 0, $\{laX\}_{X=1..5}$ avec max de 0.5, 1 et 2 et $\{laX\}_{X=6..8}$ avec max de 0.5, 1, 2 et 10, ainsi que les $carY$ avec max de 0, 0.5, 1 et 2). Les résultats obtenus montrent que concernant l'heuristique pour l'initialisation de la solution, D décroissante fournit les meilleurs résultats. Les autres résultats n'ont aucune tendance affirmée sachant que nos propositions obtiennent les meilleures solutions connues (BKS) pour $la11$, $la13$ et $la15$ avec un max de 0.25 et 0.5 ainsi que pour $la14$ avec max de 0.25, 0.5, 1, 2, 3 et 5 par rapport aux BKS obtenues par ILOG-Scheduler. Nous améliorons aussi BKS de [1] pour $la18$ avec max égal à 0 (instance *no-wait*). Les BKS restent partagées entre [1], ILOG-Scheduler et notre proposition sans aucune régularité.

Le travail effectué a permis d'adapter la méthode à base de divergence CDS aux problèmes d'ordonnancement avec time lags. Les résultats obtenus nous incitent désormais à étudier l'impact de CDS associé à des techniques classiques en ordonnancement comme les heuristiques d'insertion pour la détermination de bornes supérieures et le réglage de règles de propagation pour les bornes inférieures.

Références

[1] A. Caumond, P. Lacomme, and N. Tchernev. A memetic algorithm for the job-shop with time-lags. *Computers and Operations Research*, 35:2331–2356, 2008.

[2] M. Milano and A. Roli. On the relation between complete and incomplete search : an informal discussion. *Proceedings CPAIOR'02*, p. 237–250, 2002.

[3] L.G. Mitten. Sequencing n jobs on two machines with arbitrary time lags. *Management Science*, 5:293–298, 1958.