Ordonnancement "Just-in-Time" par Programmation par Contraintes

Jean-Noël Monette[1], Yves Deville[1] et Pascal Van Hentenryck[2]

[1] INGI-ICTEAM, UCLouvain, 1348 Louvain-la-Neuve, Belgium
[2] Brown University, Box 1910, Providence, RI 02912
{jean-noel.monette,yves.deville}@uclouvain.be, pvh@cs.brown.edu

Mots-Clés : *ordonnancement, contraintes, just-in-time job-shop, contrainte globale*

Introduction. Les problèmes d'ordonnancement avec la minimisation des coûts d'avance et de retard (earliness and tardiness) sont de plus en plus étudiés. Nous nous sommes penchés sur le problème appelé Just-In-Time Job-Shop (JITJSP) qui est un problème de Job-Shop où chaque activité a une date à laquelle elle doit se finir et des coûts linéaires d'avance et de retard si l'activité se finit avant ou après la date fixée. L'objectif est de minimiser la somme de tous les coûts. Ce problème a été étudié dans [1] qui présente des relaxations lagrangiennes et une recherche locale. Il a aussi été abordé dans [3] qui propose une recherche par voisinage large auto-adaptatif.

La programmation par contraintes est très utile pour résoudre des problèmes d'ordonnancement mais dans le cas du JITJSP, la minimisation d'une somme permet de faire peu de réduction des domaines à partir de la borne supérieure sur le coût. Nous proposons donc une contrainte globale qui prend en compte la fonction objectif et l'ensemble des contraintes de précédences pour produire un filtrage amélioré.

Une contrainte globale. L'idée principale de l'algorithme de filtrage est de résoudre un problème relâché où les contraintes de capacité des ressources sont supprimées et de voir comment le coût augmente quand la solution du problème relâché est perturbée. La relaxation correspond à un problème de PERT avec coût d'avance et de retard. Ce problème peut être résolu en temps polynomial en utilisant l'algorithme présenté dans [2]. Si la solution optimale de la relaxation respecte les contraintes de capacités, le problème est résolu.

Dans le cas contraire, il est possible de filtrer les domaines en considérant la façon dont le coût augmente si une activité est déplacée plus tôt ou plus tard dans le temps. Cette variation définit une fonction de l'instant de fin d'une activité A qui est convexe et linéaire par morceaux. Nous appellerons cette fonction Δ_A. Lorsque la fonction Δ_A dépasse la borne supérieure du coût, cela signifie que l'activité ne peut être exécutée à ce moment. Il est assez coûteux de calculer exactement Δ_A. C'est pourquoi nous proposons une borne inférieure de cette fonction, en ne considérant qu'un sous-ensemble des points de cassure (il s'agit d'une fonction linéaire par morceaux). Il est ensuite facile de calculer les endroits où Δ_A dépasse la borne supérieure du coût et de réduire le domaine de la variable à ces endroits.

Detection de précédences. En plus du filtrage ci-dessus, si la solution optimale de la relaxation ne respecte pas les contraintes de disjonction, il est aussi possible de détecter des précédences entre activités. Il y a un conflit entre deux activités si elles doivent utiliser la même machine et qu'elles se chevauchent dans le temps, dans la solution de la relaxation. Chaque conflit entre deux activités A et B peut être résolu en imposant que A vienne avant B ou que B vienne avant A. On peut calculer le coût minimum de forcer A à venir avant B en utilisant les fonctions Δ_A et Δ_B. Précisément, l'augmentation est égale à $\min_x(\Delta_A(x) + \Delta_B(x + dur(B)))$, où dur(B) est la durée de l'activité B. Si ce coût est plus large que la borne supérieure, on peut poster la précédence inverse (que B vienne avant A). De même, si le coût induit lorsque B vient avant A est plus grand que la borne supérieure, on ajoute la contrainte que A vient avant B.

L'information sur les coûts induits par les précédences permet aussi de guider efficacement la recherche en branchant sur les conflits induisant le plus grand coût (principe first-fail) et en choisissant la branche qui a le plus petit coût (principe first-success).

Méchanismes supplémentaires. Nous proposons aussi d'étendre la recherche par programmation par contraintes avec une recherche locale gloutonne sur chacune des solutions trouvées pendant le branch-and-bound. Cela permet d'améliorer rapidement la borne supérieure et donc d'avoir une meilleure propagation. De plus, nous transformons la recherche complète en recherche par voisinage large (LNS) afin de pouvoir résoudre efficacement les instances les plus grandes.

Resultats. Nous avons implémenté le propagateur et la recherche par dessus la couche CP/Scheduling de COMET. Une étude expérimentale montre que cette approche est compétitive avec d'autres recherches de l'état de l'art ([1] and [3]). En particulier, nous avons été capable d'améliorer les meilleures solutions connues pour 29 des 72 instances introduites par [1]. Nous sommes actuellement en train d'étendre l'approche pour résoudre des problèmes avec des ressources dont la capacité est plus grande que 1 (RCPSPET).

Ce travail a été présenté à ICAPS2009 et publié dans [4].

Références

[1] Philippe Baptiste, Marta Flamini, and Francis Sourd. Lagrangian bounds for just-in-time job-shop scheduling. *Comput. Oper. Res.*, 35(3) :906–915, 2008.

[2] Philippe Chrétienne and Francis Sourd. Pert scheduling with convex cost functions. *Theor. Comput. Sci.*, 292(1) :145–164, 2003.

[3] Philippe Laborie and Daniel Godard. Self-adapting large neighborhood search : Application to single-mode scheduling problems. *In Proceedings MISTA-07, Paris*, pages 276–284, 2007.

[4] Jean-Noël Monette, Yves Deville, and Pascal Van Hentenryck. Just-in-time scheduling with constraint programming. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *ICAPS*. AAAI, 2009.