Programmation par Contraintes pour planifier les déplacements de chariots dans un atelier

Valentin Antuori[1,2], Emmanuel Hébrard[2], Marie-José Huguet[2],
Siham Essodaigui[1], Alain Nguyen[1]

[1] Renault, France
{valentin.antuori, siham.essodaigui, alain.nguyen}@renault.com
[2] LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
{vantuori, hebrard, huguet}@laas.fr

Mots-clés : *Programmation par Contraintes, Ordonnancement, Voyageur de Commerce.*

1 Introduction et description du problème

Nous nous intéressons au transport de pièces dans un atelier d'assemblage du secteur automobile, de leurs points de production vers leurs points de consommation. Chaque type de pièces de l'atelier est transporté dans un chariot adapté, à capacité limité. Lorsqu'un chariot est plein, un opérateur se charge de le transporter, en formant un train de plusieurs chariots. Les cadences de production et de consommation d'une même pièce sont identiques, et donc au moment où un chariot est plein au poste de production, un chariot s'est vidé au poste de consommation correspondant. L'opérateur est également chargé de ramener les chariots vides de leurs points de consommation vers leurs points de production. Le train de chariots composé par l'opérateur peut contenir des chariots correspondant à différents types de pièces, vides ou pleins. De plus, la longueur maximale du train de chariots est limitée. La cadence de production des machines et la capacité de chaque chariot permettent de définir le cycle de production pour chaque paire de machines. Un cycle de production correspond au temps mis par la machine productrice (resp. consommatrice) pour remplir (resp. vider) un chariot. Le but est d'amener le chariot plein (resp. vide) à son poste de consommation (resp. production), avant la fin du cycle courant. Il y a ainsi pour chaque paire de machines une périodicité dans les besoins en chariots. A l'état initial , on a un chariot plein et un chariot vide devant chaque machine.

Le problème consiste donc à planifier les déplacements d'un véhicule de capacité limitée, pour satisfaire des requêtes de collectes et de livraisons avec contraintes de fenêtre de temps et périodicité des requêtes (la période est dans la plupart des cas différente pour chaque paire de machines), jusqu'à un horizon donné, sans objectif à minimiser/maximiser. Ce problème se place dans la famille des problèmes de *pickup and delivery*, à un seul véhicule et avec fenêtre de temps[4, 1]. La particularité du problème étudié se situe dans la périodicité des requêtes. On a pour chaque paire de machines, deux requêtes simultanées (transporter le chariot plein, et transporter le chariot vide), qui vont se répéter jusqu'à la fin de l'horizon temporel. Ce problème peut être aussi abordé sous le prisme de l'ordonnancement. On a alors un problème d'ordonnancement disjonctif à une machine (l'opérateur) où une tâche représente une action (prendre ou déposer un chariot), avec fenêtre de temps, temps de préparation dépendant de la séquence (pour les temps de trajets), précédences entre certaines tâches et une contrainte sur la longueur maximale du train de chariots.

2 Modélisation par contraintes

Pour résoudre ce problème, on propose de comparer deux modèles de programmation par contraintes. Un modèle type voyageur de commerce inspiré de [2], et un modèle plus léger,

inspiré de ceux utilisés en ordonnancement. Dans les deux cas, pour chaque paire de machines et pour chaque période, il y a 4 opérations : prendre/livrer un chariot plein/vide. Des précédences doivent être respectées entre une opération de collecte et l'opération de livraison associée.

Dans le modèle d'ordonnancement, pour chaque opération, on définit une variable indiquant sa date de début. De plus, pour chaque paire d'opérations, on considère une variable booléenne représentant l'ordre relatif entre ces deux opérations. On lie ces deux ensembles de variables comme suit : si l'opération i précède l'opération j, alors l'opération j doit commencer après l'opération i (en intégrant la durée de l'opération et le temps de transport). Pour respecter la contrainte sur la longueur du train, on utilise la contrainte *balance* proposée dans [3]. Nous notons ici qu'il n'est pas nécessaire de déclarer une variable pour chaque paire de tâches, et le nombre de variables peut être grandement réduit grâce à la structure du problème.

L'autre modèle de type voyageur de commerce, s'appuie sur des variables $suiv_i$ et pos_i pour chaque opération i, qui indique respectivement l'opération suivant directement l'opération i et la position de i dans la tournée. On a également les variables d'ordre pour chaque paire d'opération. Par rapport au modèle présenté dans [2] et parce que nous n'avons pas d'objectif de minimisation de coût, on remplace la contrainte *WeightedCircuit* par la contrainte *Circuit*, pour assurer le circuit hamiltonien. Pour gérer la contrainte de capacité, on ajoute une variable pour chaque opération i, qui représente la longueur du train lorsqu'on effectue l'opération i. La longueur du train de $suiv_i$ est alors égale à la longueur du train de i à laquelle on ajoute la longueur du chariot de l'opération i (négative dans le cas de la livraison).

3 Premiers résultats et travaux futurs

Dans le but de comparer les modèles, et en plus des données industrielles, un jeu d'instances a été généré aléatoirement avec différents horizons temporels et différentes classes de difficulté. Les instances aléatoires vont de 100 à 600 opérations et les instances industriels de 500 à 5000 opérations. Les deux modèles ont été implémenté à l'aide du solver Choco 4[5].

On observe que les meilleurs résultats sont obtenus avec une heuristique d'instanciation de variables qui traite le problème chronologiquement, c'est à dire qui construit la séquence dans l'ordre des visites pour le modèle TSP, ou qui séquence les opérations par dates de début au plus tôt pour le modèle d'ordonnancement. Cela est notamment dû à la structure temporelle du problème. On note que le modèle d'ordonnancement résout plus d'instances, et plus rapidement sur les instances aléatoires que le modèle TSP, qui ne passe pas à l'échelle sur les instances industrielles. Cela s'explique par la lourdeur du modèle, notamment les contraintes *Element* et *Circuit* font baisser drastiquement le nombre de noeuds explorés par seconde.

En guise de perspectives, nous nous penchons actuellement sur l'intégration de ces modèles dans un schéma de type *Large Neighborhood Search* en vue d'améliorer les résultats, de nombreuses instances restant encore non résolues. Une autre perspective concerne le problème industriel global, qui est le dimensionnement des équipes de l'atelier.

Références

[1] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems : a classification scheme and survey. *TOP*, 15(1) :1–31, Jul 2007.

[2] S. Ducomman, H. Cambazard, and B. Penz. Alternative Filtering for the Weighted Circuit Constraint : Comparing Lower Bounds for the TSP and Solving TSPTW. In *AAAI*, 2016.

[3] P. Laborie. Algorithms for propagating resource constraints in ai planning and scheduling : Existing approaches and new results. *Artificial Intelligence*, 143(2) :151 – 188, 2003.

[4] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2) :81–117, Jun 2008.

[5] C. Prud'homme, J.-G. Fages, and X. Lorca. *Choco Documentation*. TASC - LS2N CNRS UMR 6241, COSLING S.A.S., 2017.