Guider une métaheuristique à partir de solutions obtenues par un Solver puissant

Quentin Perrachon, Alexandru Liviu Olteanu, Marc Sevaux

Guider une métaheuristique à partir de solutions obtenus par un Solver puissant

Quentin Perrachon[1,2],Alexandru-Liviu Olteanu[1],Marc Sevaux[1]

[1] Lab-STICC, UMR 6285, CNRS, Equipe Decide, Université Bretagne Sud
{quentin.perrachon,alexandru.olteanu,marc.sevaux}@univ-ubs.fr
[2] HERAKLES

Mots-clés : *flexible job shop,multi-objective optimisation,industrial problem, metaheuristics,constraint programming*

1 Contexte

La société Hérakles développe et distribue un ERP-GPAO partout en France. Sa clientèle est principalement composée d'industries de très petite, petite et moyenne taille. Hérakles souhaite proposer et fournir des solutions intelligentes d'ordonnancement à ses clients. Dans le cadre d'une thèse CIFRE en collaboration entre la société Hérakles et l'équipe DECIDE du laboratoire Lab-STICC, nous présentons donc une première méthode de résolution pour des problèmes d'ordonnancement d'atelier industriel.

2 Modèle

Le problème auquel nous nous intéressons est un Job Shop avec plusieurs ressources nécessaires par opération et flexibilité des ressources (Flexible Job Shop Multi-Resource)[1]. De plus des contraintes de disponibilités sont présentes sur les différentes ressources. Toutes les ressources nécessaires pour une opération ne sont pas nécessaires durant tout le long du temps de traitement de l'opération. Une ressource ne peut être nécessaire à une opération qu'au début ou à la fin d'une opération pendant une durée inférieure à son temps de traitement totale (réglage/contrôle).[2] Le cheminement des opérations d'un job donné n'est pas forcément linéaire. Chaque job ne peut commencer qu'après sa date de disponibilité et on souhaite terminer nos jobs avant leur date échue. Cette modélisation assez générale nous permet de regrouper beaucoup des problèmes de nos clients que nous rencontrons. Nous pouvons modéliser opérateurs, machines, et même outils via des ressources différentes ainsi que des contraintes d'affectation et de disponibilité unique à chaque type de ressources.

3 Programmation par contraintes

Plusieurs métaheuristiques à base de voisinage et de recherche locale ont été développées et testées sur des instances de notre problème. Peu de travaux existent sur des problèmes de job shop flexible multi-ressources et il est difficile de comparer nos résultats avec ceux-ci. Nous avons donc comparé nos méthodes avec diverses autres méthodes utilisées régulièrement pour des problèmes d'ordonnancement. Notamment des méthodes exactes dans le but d'avoir une idée générale de l'efficacité de nos méthodes. La programmation linéaire ne donne pas de résultats intéressants en un temps convenable. Cependant, une modélisation en programmation par contrainte et l'utilisation du Solver CP Optimizer[3] nous permet d'obtenir des solutions très intéressantes en des temps record comparées à une métaheuristique simple de type récuit simulé (TAB. 1).

Instance	Nombre d'opérations	Nombre de ressources	Résultats SA	Résulats CPP
1	25	6	**444***	**444***
2	50	9	**463***	**463***
3	100	15	5404	**3992**
4	100	15	2806	**2303***
5	200	21	18125	**14439**
6	200	21	13840	**7993**
7	300	21	41555	**32834**
8	300	21	44539	**36151**

TAB. 1 – Comparaison des résultats métaheuristiques et CPP(CP Optimizer)
Pour toute les instances : chaque opération requièrt une à deux resources avec flexilibilité partielle
Les deux méthodes sont lancées avec le même budget cpu et temps
L'objectif considéré est la somme des retards

Ces instances ont été générées à partir d'un générateur que nous avons développé dans un but d'avoir des instances ressemblant à des situations réelles. La différence entre les résultats des deux méthodes augmente avec la taille des instances à l'avantage de la CPP. Des résultats annexes nous montrent aussi que la contrainte multi-ressource du problème complexifie grandement l'espace des solutions en taille mais aussi topologiquement avec beaucoup plus d'optimum locaux et de plateaux.

4 Perspective

L'utilisation d'un solver tel que CP Optimizer n'est pas envisageable commercialement, mais ces résultats nous montrent que nous avons encore une marge d'amélioration importante sur les solutions obtenues à partir des métaheuristiques. Nous souhaitons donc comprendre les différences entre les solutions obtenues par nos métaheuristiques et la programmation par contraintes et pourquoi les métaheuristiques développées n'explorent pas dans la direction des solutions obtenus par la PPC et potentiellement remédier aux problèmes et améliorer nos résultats.

Références

[1] Imran Ali Chaudhry and Abid Ali Khan. A research survey : review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3) :551–591, 2016.

[2] Stéphane Dauzère-Pérès and Claire Pavageau. Extensions of an integrated approach for multi-resource shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 33(2) :207–213, 2003.

[3] Philippe Laborie, Jérôme Rogerie, Paul Shaw, and Petr Vilím. Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2) :210–250, 2018.