Recherche arborescente de Monte-Carlo pour un problème industriel de collecte et livraison de pièces

Valentin Antuori[1,2], Emmanuel Hébrard[2,3], 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
[3] ANITI, Université de Toulouse, France

Mots-clés : *Recherche arborescente de Monte-Carlo, Ordonnancement, Voyageur de Commerce.*

1 Introduction

La recherche arborescente de Monte-Carlo [3](MCTS, Monte-Carlo Tree Search) a connu un développement important, en particulier dans le domaine des jeux. Le principe de cette méthode est de guider l'expansion d'un arbre de recherche par des estimations faites par des tirages de Monte-Carlo, tout en maintenant un compromis entre exploitation et exploration.

En optimisation combinatoire, l'exploration exhaustive d'un arbre de recherche semble dépasser les capacités des approches actuelles. En effet, dans de nombreux cas, l'espace de recherche ne peut pas être suffisamment réduit par des méthodes exactes, et ces méthodes ne passent pas à l'échelle. La méthode MCTS offre une stratégie générique pour s'attaquer à ces problèmes en ne nécessitant que très peu de connaissances spécifiques [2, 4].

Nous proposons d'appliquer cette méthode à un problème de collecte et de livraison de pièces avec un véhicule, issu d'un atelier d'assemblage dans le contexte de la construction automobile. Chaque type de pièces dispose de sa propre cadence de collectes et livraisons qui doivent être répétées sur un horizon de temps donné. Les opérations de collecte et livraison doivent ainsi respecter des fenêtres de temps, de plus la capacité du véhicule est limitée. La description complète du problème est présentée dans [1].

Le principe de la méthode MCTS repose sur la répétition de 4 phases. La phase de *sélection* permet de parcourir l'arbre de la racine jusqu'à un noeud non ouvert via un mécanisme de bandit manchot à chaque étape. Chaque noeud contient les résultats des précédentes itérations et le nombre de fois où il a été visité par cette phase. Le principe est de sélectionner le noeud le plus prometteur au regard des précédentes itérations, tout en accordant un bonus qui décroît à chaque visite pour encourager l'exploration. Ensuite, la phase d'*expansion* permet d'ajouter un ou plusieurs fils à ce noeud sélectionné. Vient ensuite la phase de *simulation*, dont le principe est d'étendre le noeud sélectionné jusqu'à une feuille de l'arbre par des tirages de Monte-Carlo. Dans le contexte de l'optimisation combinatoire, cette procédure est généralement faite par une heuristique. Enfin, le résultat de cette simulation est *retro-propagé* : les noeuds traversés lors de la phase de simulation sont mis à jour (valeur, nombre de visites) pour prendre en compte cette simulation lors des prochaines phases de sélection.

2 Contributions

L'objectif du problème est de trouver une séquence contenant toutes les opérations, qui minimise le retard maximal (on rappelle qu'il n'y a qu'un seul véhicule). Si ce retard est nul,

alors l'instance est résolue. Dans la méthode MCTS, un noeud de l'arbre est une sous-séquence d'opérations, et un enfant de ce noeud correspond à cette même séquence avec une opération supplémentaire. Lors de la phase d'expansion, on ajoute un enfant pour chaque action possible. La phase de simulation utilise une heuristique stochastique apprise par apprentissage par renforcement issu de travaux précédents [1]. Nous proposons trois adaptations de la méthode.

Évaluation. La méthode vise à obtenir une évaluation des noeuds, et donc des décisions, en se basant sur les simulations des itérations précédentes. Une simulation étant une procédure gloutonne, on dispose pour chaque décision de l'accroissement marginal de la borne inférieur sur le retard maximal, et on peut conjecturer qu'à mesure que la procédure avance, la corrélation entre la qualité de la décision initiale, et l'accroissement de cette borne inférieure diminue. Nous proposons alors d'évaluer un noeud par l'espérance de la somme des accroissements marginaux de la borne inférieure obtenus après ce noeud, pondéré par un coefficient décroissant exponentiellement.

Compromis dynamique. Dans une méthode MCTS, le compromis entre exploitation et exploration est fait via un paramètre dans la phase de sélection. Nous proposons une adaptation dynamique de ce paramètre en fonction de la profondeur de l'arbre, de manière à forcer l'exploitation dans la partie haute de l'arbre au fur et à mesure de sa croissance. Ce compromis dynamique produit un effet similaire à un mécanisme de "commit" souvent utilisé avec cette méthode.

Recherche en profondeur. Enfin, nous proposons d'utiliser une méthode de Depth First Search (DFS) à budget limité durant la simulation. Plus précisément, la procédure gloutonne est remplacée par une DFS, et un budget (pouvant être nul) de *backtrack* est accordé lors du premier échec. Le budget est dépendant de la qualité de cette première branche (du noeud sélectionné, jusqu'à l'échec). Lorsque le budget est totalement dépensé, la meilleure branche obtenue, si elle n'est pas complète, est étendue via la procédure gloutonne.

Les évaluations expérimentales de la méthode, en conjonction de ces trois mécanismes, sont réalisées sur un ensemble d'instances générées aléatoirement sur la base des caractéristiques du problème industriel. La méthode nous permet de résoudre près de 10% d'instances supplémentaires, et d'améliorer la fonction objectif d'environ 10% également, par rapport aux résultats antérieurs obtenus par programmation par contraintes ou par une méthode de recherche locale. Plus précisément, on observe que la DFS et le compromis dynamique agissent sur le nombre d'instance résolue, tandis que la fonction d'évaluation proposée agit sur la valeur de l'objectif.

Des travaux futurs visent à étudier la généricité des mécanismes proposés en élargissant l'étude à d'autres problèmes d'optimisation combinatoire.

Références

[1] Valentin Antuori, Emmanuel Hebrard, Marie-José Huguet, Siham Essodaigui, and Alain Nguyen. Leveraging Reinforcement Learning, Constraint Programming and Local Search : A Case Study in Car Manufacturing. In *Principles and Practice of Constraint Programming (CP)*, pages 657–672, 2020.

[2] Dimitris Bertsimas, J. Daniel Griffith, Vishal Gupta, Mykel J. Kochenderfer, and Velibor V. Misic. A comparison of Monte Carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems. *European Journal Operational Research*, 263(2) :664–678, 2017.

[3] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1) :1–43, 2012.

[4] Minh Anh Nguyen, Kazushi Sano, and Vu Tu Tran. A Monte Carlo Tree Search for Traveling Salesman Problem with Drone. *Asian Transport Studies*, 6 :100028, 2020.