Adaptive Large Neighbourhood Search pour un problème appliqué de Pickup and Delivery avec fenêtres de temps *

Matthieu Fagot [1,2], Laure Brisoux Devendeville [1], Corinne Lucet[1]

[1] Laboratoire MIS (UR 4290), Université de Picardie Jules Verne, France
{laure.devendeville, corinne.lucet}@u-picardie.fr
[2] Smile Pickup, France
mfagot@smilepickup.com

Mots-clés : *Routage, Pickup and Delivery Problem, Fenêtres de temps, ALNS*

1 Introduction

L'entreprise Smile Pickup gère un réseau de points relais pour gros colis ainsi que la planification du transport des colis entre les magasins et les points relais. La planification du transport de Smile Pickup se rapproche des problèmes de *Pickup and Delivery*[1] avec fenêtres de temps (PDPTW) auxquels on ajoute des contraintes spécifiques à l'activité de l'entreprise. Le problème comporte trois catégories de points : les dépôts de véhicules, les magasins et les points relais. Ces points peuvent avoir plusieurs fenêtres de temps sur une même journée. La flotte de véhicules est hétérogène et doit livrer des palettes de colis depuis les magasins jusqu'aux points relais. Des pénalités croissantes sont déduites de l'objectif pour chaque jour de stockage supplémentaire des palettes en magasin. L'objectif est d'optimiser les critères suivants : la distance parcourue, le nombre de véhicules utilisés, les pénalités de stockage et le nombre de palettes non livrées. Pour l'entreprise, ces objectifs permettent de concilier rentabilité économique et qualité de service. La spécificité du problème réside dans le fait que les tournées puissent passer plusieurs fois par les mêmes points et que chaque point puisse être à la fois le départ et la destination de plusieurs palettes. À notre connaissance, il n'existe, à ce jour, aucune publication traitant cette dernière contrainte directement sans passer par une duplication des points et ainsi se ramener à un problème classique de PDPTW. Cette solution est en pratique peu viable compte tenu du nombre de palettes traitées. Dans cet article, nous présenterons brièvement l'algorithme de recherche locale que nous avons mis en place puis nous évaluerons les performances de celui-ci.

2 L'Adaptative Large Neighbourhood Search

L'algorithme Adaptative Large Neighbourhood Search(ALNS) [2] est une amélioration de l'algorithme de Large Neighbourhood Search (LNS) dans lequel des phases de détérioration et d'amélioration de la solution sont enchaînées afin d'explorer l'espace des solutions. Durant ces phases, des mouvements sont choisis selon une distribution de probabilités puis appliqués à la solution afin d'en explorer le voisinage. Ces phases sont regroupées en segments à l'issue desquels ces probabilités sont mises à jour. L'ALNS diffère du LNS en permettant l'adaptation automatique de la distribution des probabilités au cours de l'exécution afin d'adapter l'utilisation des mouvements au paysage de la fitness. Cette technique s'apparente à de l'apprentissage par renforcement.

*Ce projet est soutenu par le projet PICKOPT (CIFRE n° 2021/0599 entre Smile Pickup et le Laboratoire MIS)

Une itération de l'ALNS permet de passer d'une solution à une solution voisine. Chaque itération se subdivise en quatre phases. La première consiste à détériorer la solution courante grâce à trois mouvements de suppression de palette, de point ou de tournée. La seconde phase permet l'amélioration de la solution dégradée par l'application de mouvements d'insertion de palettes jusqu'à la saturation de la nouvelle solution. Lors de la phase suivante, si la nouvelle solution est améliorante ou si celle-ci est acceptée par le critère d'aspiration, elle sera utilisée comme nouvelle solution courante lors de la prochaine itération. Une méthode de Simulated Annealing (SA) est utilisée comme critère d'aspiration. Celle-ci accepte certaines solutions détériorantes pour échapper aux optima locaux. La probabilité d'accepter une nouvelle solution S_p dépend de la solution courante $S : p(S_p) = e^{-\frac{f(S)-f(S_p)}{T}}$. La température T contrôle la notion de proximité des scores : plus T est élevée, plus la différence des scores est tolérée. Deux scénarios ont été testés. Le premier décrémente la température par un facteur multiplicatif à intervalle régulier. Le second permet à l'algorithme de revenir à une température élevée si aucune amélioration n'a été trouvée pendant un certain nombre d'itérations.

La dernière phase a pour but de mettre à jour les probabilités W_m^s associées à chaque mouvement m à la fin de chaque segment $s : W_m^{s+1} = \rho W_m^s + (1-\rho)\frac{\pi_m}{\theta_m}$ avec $\rho \in]0, 1[$. π_m est le score du mouvement m obtenu durant le dernier segment et θ_m le nombre de fois où celui-ci a été utilisé dans ce même segment [2].

Par ailleurs une mémoire taboue a été utilisée. Lorsqu'une palette est retirée d'une tournée, nous interdisons qu'elle soit ré-insérée dans celle-ci avant un certain nombre d'itérations. De plus nous avons testé plusieurs politiques pour la phase de détérioration : (1) une seule opération de suppression par phase, (2) une suppression d'un taux minimum de palettes. Plusieurs scénarios ont été étudiés comme l'utilisation d'un pourcentage constant, d'un pourcentage décroissant au cours de l'exécution ou une réinitialisation puis une augmentation progressive de celui-ci après l'amélioration de la meilleure solution.

3 Résultats

Nous avons testé nos algorithmes sur un ensemble d'instances que nous avons généré aléatoirement en essayant de rester proche de la réalité de l'entreprise. Les premiers résultats montrent que l'utilisation du critère d'aspiration *SA* sur les différents algorithmes augmente considérablement les performances avec un gain de près de 8% en moyenne. La différence entre les deux scénarios de gestion de la température montre que l'ajout de *restart* apporte un léger gain ($\sim 1\%$) sur les instances de grande taille. Pour les petites instances, les solutions optimales sont trouvées rapidement dans la majorité des algorithmes.

4 Perspectives

Nos recherches vont maintenant se tourner vers l'analyse des performances de notre algorithme afin de décider si l'utilisation d'algorithme plus complexe comme les métaheuristiques peut être utile à l'entreprise. Nous allons aussi continuer à étudier de nouveaux scénarios et modules que nous pourrions apporter à l'algorithme notamment l'utilisation d'une méthode d'ϵ-greedy pour gérer le choix des mouvements. Pour finir, l'ajout de nouvelles contraintes est envisagé afin de se rapprocher encore plus de la réalité de l'entreprise.

Références

[1] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems : a classification scheme and survey. *Top*, 15(1) :1–31, 2007.

[2] David Pisinger and Stefan Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer, 2010.