Pilotage Dynamique de Transport Sanitaire: Apprentissage et Optimisation

Omar Rifki, Garaix Thierry

Pilotage Dynamique de Transport Sanitaire: Apprentissage et Optimisation

Rifki Omar [1], Garaix Thierry [1]

[1] Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CIS, Departement I4S, F-42023. E-mails : {omar.rifki, garaix}@emse.fr

Résumé. Nous proposons deux méthodes de résolution pour le transport dynamique, *i.e.* arrivage des requêtes en temps réel, suivant une synergie entre apprentissage automatique et optimisation. La méthode d'optimisation de base est une ré-optimisation selon [Bertsimas *et al*., 2019]. La première amélioration utilise un réseau de neurones pour exploiter le contexte des instances à résoudre et produire des temps de ramassage de points réalistes. Tandis que la deuxième amélioration fait usage de l'apprentissage par renforcement en plus de marches aléatoire pour la construction du graphe support pour l'instance du problème correspondant à chaque pas de temps. Les benchmarks considérés dans le projet sont ceux de [Bertsimas *et al*., 2019] pour le transport urbain et de [Skiredj, 2021] pour le transport sanitaire.

Mots clés: transport sanitaire, requêtes dynamiques, apprentissage, réseaux de neurones, Q-learning.

Introduction

Le transport sanitaire consiste à transporter depuis et/ou vers des pôles de santé des patients pour des raisons de soins et de diagnostic. En France, les ambulanciers effectuent plus de 40 millions de voyages pour transporter plus de 5 millions de patients chaque année [CNSA, 2018]. La spécificité du transport sanitaire se trouve dans les différentes contraintes de ce problème : i/ type de véhicules, *e.g.* véhicule sanitaire léger, ambulance, taxi, sachant que le tarif varie selon le type de véhicule et que certains patients nécessite un type spécifique de véhicule, ii/ prise en compte de l'urgence des requêtes, *i.e.* certains patients en vu de leur état de santé sont plus prioritaires que d'autres, iii/ contraintes horaires liées aux régulations des ambulanciers, *e.g.* les heures de pause et de travail, iv/ nature des requêtes : certaines requêtes sont connues à l'avance mais pas d'autres, etc. Ce secteur du transport sanitaire est actuellement en pleine croissance dû au vieillissement de la population et à l'augmentation des maladies chroniques. En plus, suite aux stratégies de rachat et de coopération entre les transporteurs sanitaires, plusieurs sociétés du domaine gèrent quotidiennement une large demande qui peut s'élever à 3000 requêtes, *e.g.* Lomaco. Pour toutes les raisons mentionnées auparavant, il est indispensable de développer des algorithmes adaptés à ce type transport capable de répondre à une demande très large, sans mentionner que l'uberisation des systèmes de transport touche également le transport sanitaire.

1 Description du problème

Le transport sanitaire appartient à la famille des problèmes d'appel d'un trajet (*Dial-a-Ride problem – DARP*) qui sont un variant du problème de tournées de véhicules (*Vehicle Routing Problem – VRP*) destinés au transport de personnes. Le DARP inclut additionnellement au VRP des contraintes de précédence, de temps maximal d'attente et de conduite. Les contraintes de fenêtres temporelles (TW) sont également imposées. Voir [Ho et

al., 2018] pour les exemples du DARP destinés au transport sanitaire. La méthode de [Bertsimas *et al.*, 2019] arrive à résoudre un cas particulier du DARP-TW dynamique (le problème de taxis) pour une grande demande : environ 600-6500 clients pour chaque ré-optimisation qui dure 30 secondes sur le réseau de Manhattan. Pour cette raison, nous considérons cette approche de résolution comme un point de départ pour tester l'intégration optimisation-apprentissage automatique (*Machine Learning – ML*) dans le contexte dynamique. Notre idée de base est d'ajouter une routine ML au sein de cette ré-optimisation pour améliorer la qualité des solutions. De plus en plus de solveurs optent pour ce type d'intégration ML-optimisation, voir l'enquête [Bengio *et al.*, 2019].

2 Approches de résolution

L'approche de [Bertsimas et al., 2019] est une ré-optimisation qui s'appuie sur une grande simplification du graphe de travail afin de rendre la résolution plus efficace pour chaque pas de temps. Tout d'abord, le graphe G est rendu acyclique en supprimant les arcs qui ne respectent pas les fenêtres temporelles (et d'autres contraintes temporelles plus strictes). Ensuite le graphe G est filtré en un graphe KG, en sélectionnant que les K meilleurs arcs entrants et sortants de chaque nœud du graphe G, avec K est un paramètre de l'algorithme. Sans élimination de sous-tours, le problème de routage devient un problème de flot maximal sur réseau (**MaxFlow**), avec des variables entières et réelles. Lorsque les temps de ramassage des clients sont fixés (en faisant abstraction des fenêtres temporelles), le problème MaxFlow devient un problème solvable efficacement par l'algorithme du simplexe (**MaxFlow-FT – Fixed Times**), donc plus facile à résoudre que MaxFlow. Chaque client c a un temps de requête $t_c^{request}$, un temps de confirmation $t_c^{conf} > t_c^{request}$, où l'optimiseur lui confirme si sa requête est acceptée ou non, et une fenêtre temporelle pour le ramassage : $I_c = \left[t_c^{min}, t_c^{max}\right]$. L'algorithme appelé à chaque ré-optimisation de [Bertsimas *et al*., 2019] est comme suit :

1. **while** (le budget temps est disponible) **do**
2. Initialiser le graphe BG avec les nœuds de G sans aucun arc ;
3. Ajouter la solution s en cours à BG ;
4. **while** (le nombre d'arcs de BG ≤ E_{max}) **do**
5. Pour chaque client c, générer uniformément un temps de ramassage $t_c \in I_c$;
6. Résoudre **MaxFlow-FT** sur KG avec les temps fixés t_c pour tout client c ;
7. Ajouter la solution de 6. aux arcs de BG ;
8. **end**
9. Résoudre **MaxFlow** sur BG ;
10. Mettre à jour la solution s courante ;
11. **end**

Dans un premier lieu, cet algorithme crée un graphe backbone BG avec un nombre d'arcs limité à E_{max} , avant de résoudre **MaxFlow** sur ce même graphe. Le paramètre E_{max} est choisit de tel sorte que **MaxFlow** est résolvable en un temps raisonnable.

Notre première routine ML remplace les étapes 2 à 9. Au lieu de générer aléatoirement des temps de passage (étape 5.) et de résoudre le problème MaxFlow-FT sur ces temps-là (étape 6.) pour la construction du graphe de support backbone (étage 7.), nous apprenons les meilleurs temps de passage t_c , et résolvons MaxFlow-FT avec ces temps-ci. Cette solution met à jour la solution courante. Concernant la routine apprentissage, nous utilisons un réseau de neurones récurrents (R*ecurrent Neural Network – RNN*) qui est un réseau de neurones qui peut mieux gérer des séquences de vecteurs en fonction du temps [Alom *et al*., 2019]. Pour l'entraînement du RNN, nous résolvons MaxFlow-FT avec plusieurs temps de passage fixés choisit aléatoirement, en plus du cas des temps égaux aux bornes supérieures des TW. La sortie du RNN es les temps de ramassage des

clients, alors que l'entrée est la solution du MaxFlow-FT correspondant tel que si un arc est dans la solution sont la valeur prise est le profit de l'arc, sinon une valeur nulle est accordée. On obtient les temps de passage finaux pour un pas de temps en prenant pour entrée les profits de tous les arcs du graphe G de la ré-optimisation en cours.

Notre deuxième routine ML remplace les étapes 4 à 8, avec des chemins aléatoires et des chemins construits suivant un apprentissage par renforcement dans le graphe KG. Les deux types de chemins partent des positions actuelles des taxis. Le paramètre p_{rl} désigne le pourcentage de taxis pour lequel le chemin est construit avec l'algorithme d'apprentissage Q (*Q-learning*) [W*atkins et Dayan, 1992*].

2.1 Résultats préliminaires

Ci-dessous quelques résultats sur les données de [Bertsimas *et al*., 2019]. Nous avons fixé le temps de simulation à 15 minutes et les temps de confirmation à 4 minutes après les temps de requêtes (si $t_c^{conf} > t_c^{min}$, nous prenons t_c^{min}). Le pas de temps est égale à 1 et à 5 minutes. Les temps de requêtes maximaux et le nombre de taxis sont variés comme dans le tableau. Les profits en terme de dollars des trajets sont calculés selon [Bertsimas *et al*., 2019], et les coûts des arcs sont proportionnels aux temps de trajets. Le paramètre p_{rl} est fixé à $p_{rl}=10\%$.

Paramètres d'entrée			Bertsimas		RNN			Q-learning		
#taxis	temps de requête maximal	Pas de temps	profits ($)	#clients non servis	profits ($)	#clients non servis	Gain profit (%)	profits ($)	#clients non servis	Gain profit (%)
4000	5min	1min	41380.1	842	27992.6	1879	-32.3	40809.6	876	-1.37
4000	5min	5min	38194.0	1140	32768.0	1633	-16.5	38769.3	1059	+1.5
4000	15min	1min	41291.3	863	34250.1	1404	-20.5	40955.7	874	-0.8
4000	15min	5min	37825.8	1188	16644.6	2848	-55	38307.7	1105	+1.3
3000	5min	1min	41301.2	897	27934.4	1921	-32.3	40805.1	921	-2.1
3000	5min	5min	37913.3	1205	16463.6	2860	-56	37948.4	1159	+0.9
3000	15min	1min	42470.1	763	30690.2	1667	-27.7	42054.4	826	-1.0
3000	15min	5min	38223.6	1155	16926.0	2829	-55.7	38394.7	1114	+0.4

Conclusion et perspectives

Dans ce résumé allongé, nous avons introduit une intégration de deux routines d'apprentissage dans l'algorithme de ré-optimisation de [Bertsimas *et al*., 2019]. Nous l'avons appliqué aux requêtes de taxis de Manhattan. Ces données sont caractérisées par une grande demande et une grande offre également (une large flotte de taxis), la résolution de ce problème de transport s'apparente plus à un problème d'affectation à cause de la grande densité du territoire d'application. L'approche Q-learning offre de meilleur résultats par rapport au réseau RNN dépassant l'approche de Bertsimas lorsque le pas de temps de la ré-optimisation est grand. Les données de santé sont souvent appliquées sur des territoires peu denses, et avec des grands pas de temps, comme dans le cas des données [Skiredj, 2021] de Lomaco pour le transport sanitaire. Nous

pensons que l'apprentissage aura un grand impact dans de cas-ci encore plus que les territoires denses, comme la composante de routage sera plus prépondérante que celle de l'affectation.

Références

Bertsimas, D., Jaillet, P. et Martin, S. (2019). *Online vehicle routing: The edge of optimization in large-scale applications*. Operations Research, 67(1), pp.143-162.

Skiredj, M. (2021). *Planification proactive et réactive des réseaux d'ambulances.* Thèse de doctorat EMSE-CIS.

CNSA (2018). *LivreBlanc Chambre nationale des services d'ambulances*. Lien: https://www.cnsa-ambulances.com/wpcontent/uploads/2019/05/Livre_blanc_CNSA.pdf.

Ho, S.C., Szeto, W.Y., Kuo, Y.H., Leung, J.M., Petering, M. and Tou, T.W. (2018). *A survey of dial-a-ride problems: Literature review and recent developments*. Trans. Research Part B: Meth., 111, pp.395-421.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). *Machine learning for combinatorial optimization: a methodological tour d'horizon*. EJOR, 290(2), 405-421.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2019). *A state-of-the-art survey on deep learning theory and architectures*. Electronics, 8(3), 292.

Watkins, C. J., & Dayan, P. (1992). *Q-learning. Machine learning, 8(3), 279-292.*