Approche décentralisée d'insertion avec amélioration continue de la qualité de la solution pour un système TAD

Alaa Daoud, Flavien Balbo, Paolo Gianessi, Gauthier Picard

Approche décentralisée d'insertion avec amélioration continue de la qualité de la solution pour un système TAD

Alaa Daoud[1,2], Flavien Balbo[1,2], Paolo Gianessi[1], Gauthier Picard[1,2]

[1] Univ Lyon, Mines Saint-Étienne,
[2] CNRS, Laboratoire Hubert Curien, UMR 5516, F-42023 Saint-Étienne, France.

alaa.daoud@emse.fr flavien.balbo@emse.fr paolo.gianessi@emse.fr gauthier.picard@emse.fr

Résumé

Dans le cadre du transport à la demande (TAD), l'appariement centralisé traditionnel permet d'obtenir des solutions optimales, mais reste NP-difficile et donc inadapté aux problèmes en ligne et dynamiques. Les approches décentralisées permettent d'obtenir des solutions réalisables en ligne, mais sans garantie de qualité. Ici, nous considérons une version dynamique du TAD, appelée Dial-A-Ride-Problem (DARP) où un réseau inter-véhiculaire est le support aux communications et les décisions sont prises de manière décentralisée par les véhicules. Dans ce cadre multi-agent, nous présentons un nouvel algorithme heuristique décentralisé d'insertion fondé sur une approche classique consistant à associer les véhicules aux demandes les plus proches dans le temps et l'espace, étendue par une phase d'optimisation afin d'améliorer la qualité de la solution. Nous évaluons ses performances sur des données synthétiques et la comparons à une approche gloutonne.

Mots-clés

TAD, DARP, V2V, décentralisation, SMA, heuristique d'insertion

Abstract

In the context of On-Demand Transport (ODT), traditional centralized matching provides optimal solutions, but remains NP-Hard and therefore unsuitable for online and dynamic problems. Decentralized approaches provide feasible solutions in real time, but without any guarantee of quality. Here, we consider a dynamic version of ODT, called Dial-A-Ride-Problem (DARP) where a V2V network is the communication support and decentralized decisions are made by vehicles. In this multi-agent framework, we present a new decentralized insertion heuristic algorithm based on a classical approach consisting in associating vehicles to the closest requests in time and space, extended by an optimization phase in order to improve the quality of the solution. We evaluate its performance on synthetic data and compare it to a greedy approach.

Keywords

On-Demand Transport (ODT), DARP, V2V, decentralized, MAS, insertion heuristic.

FIGURE 1 – Un exemple de problème AVFAP, avec les sources de demande (triangles) et les taxis (cercles) avec leurs portées de communication respectives (en bleu).

1 Introduction

Le concept de transport à la demande (TAD) a été formulé pour la première fois vers 1990 comme une solution à la désaffection croissante des utilisateurs potentiels des transports publics, en particulier la nuit. La recherche dans ce domaine se concentre principalement sur l'allocation des véhicules aux demandes, le routage et la gestion de flotte.
L'affectation d'un ensemble de ressources à un ensemble d'agents est l'un des problèmes d'optimisation combinatoire les plus fondamentaux des plus étudiés dans la littérature [Danassis et al., 2019]. Il peut être résolu par programmation linéaire car sa relaxation de la formulation admet des solutions intégrales optimales à un instant donné t. Cependant, mise en oeuvre par un *dispatcher* central, cette approche, exige que les véhicules aient un accès continu à leur portail via l'infrastructure de communication globale (4G, par exemple), ce qui est coûteux et peut faire du portail un goulot d'étranglement critique. La complexité algorithmique du problème, qui étend le problème du voyageur de commerce (TSP), fait qu'il est difficile pour un *dispatcher* centralisé de gérer la dynamique du problème pendant son exécution (demandes en continue, problèmes de circulation et autres dynamiques de l'environnement). On peut s'attendre à ce que la décentralisation permette de faire face à ces problèmes.

L'heuristique d'insertion de Solomon [1987] est une méthode populaire pour résoudre une variété de problèmes d'ordonnancement et de routage. Elle peut être utilisée comme une méthode pour trouver rapidement une solution réalisable mais sans garantie de qualité de la solution. Dans le cas du problème de routage des véhicules (VRP), elle produit une solution en insérant de façon répétée des demandes non programmées dans un itinéraire partiellement construit ou comme première demande dans un nouvel itinéraire.

Dans ce travail, nous proposons le problème d'allocation de flotte de véhicules autonomes (AVFAP), qui étend le DARP traditionnel en considérant une flotte de véhicules autonomes qui se coordonnent sans *dispatcher* central. Les véhicules prennent des décisions décentralisées selon des informations échangées via une communication pair-à-pair (P2P). La communication véhicule-à-véhicule (V2V) via la communication dédiée à courte distance (DSRC) offre une faible latence, une connectivité réseau rapide, et une portée de communication allant jusqu'à 300 m [Dey et al., 2016]. Avec des portées de communication limitées (illustrée Figure 1), chaque véhicule n'est conscient que d'un sous-ensemble de demandes, et il ne communique avec une autre entité (un véhicule ou une source de demande) que si elle est directement à portée ou par transitivité. Nous proposons dans ce travail une nouvelle heuristique décentralisée pour l'AVFAP, bénéficiant de la forte réactivité des heuristiques d'insertion, et des bons résultats obtenus par l'optimisation par l'échange des demandes entre véhicules.

Cet article est organisé comme suit. Dans la section 2, nous donnons un aperçu des efforts déployés dans la littérature pour résoudre ce problème. La section 3 expose notre définition du problème AVFAP. Ensuite, nous présentons notre solution pour assurer une prise de décision rapide par les agents dans la section 4 et notre proposition d'amélioration en continu est détaillée dans la section 5. Les paramètres des expérimentations, l'évaluation et les résultats sont détaillés dans la section 6. Enfin, nous concluons le document dans la section 7 avec quelques perspectives.

2 Travaux connexes

Les approches traditionnelles du TAD envisagent soit une architecture avec *dispatcher* centralisé comme dans [Egan et Jakob, 2015; Shen et Lopes, 2015] soit une architecture multi-agent (SMA) décentralisée pour réduire la complexité des problèmes comme dans [El Falou et al., 2014; Grau et al., 2013]. Dans ce travail, nous considérons cette seconde famille de travaux. Parmi les nombreuses approches pour les décisions décentralisées et l'auto-coordination, Glaschenko et al. [2009] introduit une solution de planification en temps réel fondée sur les offres de plusieurs agents dans des environnements entièrement décentralisés, dans laquelle chaque véhicule est représenté par un agent qui peut négocier par des canaux radio avec des critères de décision flexibles. Un algorithme de reconnaissance des formes est utilisé pour prédire les emplacements les plus probables pour la prochaine demande pour recommander les déplacements vers ces emplacements.

Étudiant l'applicabilité de la programmation génétique (GP) pour développer des SMA décentralisés qui résolvent le DARP. van Lon et al. [2012] présente une méthode pour générer automatiquement un SMA qui peut résoudre le DARP pour un ensemble spécifique de scénarios. La programmation génétique est utilisée pour générer une heuristique qui est efficace pour résoudre le DARP. Le meilleur résultat obtenu avec cette approche est la planification d'une seule demande à l'avance par véhicule, ce qui maximise les intérêts locaux de l'agent et produit une solution réalisable très rapidement.

Grau et Romeu [2015] ont proposé un modèle d'agent où l'appariement de l'offre et de la demande dépend des événements simulés du marché réel des taxis. Selon leur conclusion, l'une des principales limites du modèle est l'hypothèse d'une distribution uniforme de la demande dans la zone de service.

Pour remédier à l'incertitude causée par la nature dynamique des demandes en ligne, et en ignorant le temps nécessaire à l'exécution d'un algorithme de planification, Agatz et al. [2011] ont proposé d'utiliser une approche déterministe de solution à horizon glissant, dans laquelle les plannings sont établis en utilisant toutes les informations connues dans un horizon de planification, qui est "glissé" vers l'avant pour inclure plus d'informations. Dans nos travaux précédents [Picard et al., 2018], sur la base de la coordination des véhicules par communication V2V, nous avons proposé le modèle d'allocation de ressources en ligne avec contraintes de communication (OLC^{2}RA). La solution proposée concerne une flotte de taxis autonomes, dans laquelle un véhicule décide de sa prochaine destination en ne considérant qu'une seule demande à l'avance. Au contraire, l'heuristique décentralisée ALMA de Danassis et al. [2019] est complètement découplée et ne nécessite pas de communication entre les participants. Ils démontrent une limite supérieure de la vitesse de convergence qui est polynomiale à la quantité désirée de ressources et d'agents concurrents par ressource ; Dans le cas réaliste où les quantités mentionnées sont limitées quel que soit le nombre total d'agents/ressources, le temps de convergence reste constant à mesure que la taille totale du problème augmente. Cependant,la détection des conflits nécessite toujours une communication avec d'autres véhicules, ressources ou une entité centrale pour permettre aux ressources de partager des informations sur leur statut, comme le mécanisme de coordination du tableau noir.

Jusqu'à présent et à notre connaissance, les efforts déployés dans le domaine de la planification DARP fournissent soit des solutions optimales avec un temps d'exécution souvent peu adapté à la dynamique, soit une solution solution réalisable (rapide est non nécessairement optimale) pour répondre aux demandes en ligne. Dans ce travail, comme la solution optimale pour ce problème dynamique est irréalisable avec un temps de calcul raisonnable, notre proposition combine les avantages de ces approches dans une heuristique qui fournit une réponse rapide fondée sur les enchères (solution rapide et faisable), puis cette solution est améliorée progressivement grâce à un protocole d'échange entre

agents des demandes dont la planification est à remettre en cause suite à l'évolution de l'environnement.

3 Modèle du problème AVFAP

Le problème de l'allocation d'une flotte de véhicules autonomes (AVFAP) étend le DARP traditionnel en considérant une flotte de véhicules autonomes qui se coordonnent sans *dispatcher* central. Les véhicules prennent des décisions décentralisées selon des informations échangées via une communication pair-à-pair (P2P). Le graphe de la carte de la ville, illustré dans la Figure 2, est constitué de nœuds qui représentent des emplacements géographiques et d'arcs qui représentent les liaisons routières entre ces emplacements. Une flotte de véhicules autonomes est répartie dans la ville, chaque véhicule possède un ensemble de propriétés dont les valeurs sont constantes (capacité, coût et vitesse moyenne) ou variables (emplacement, planning) car dépendantes du temps. Les passagers émettent des demandes à partir de différents endroits (que nous appellerons sources ensuite). Chacune prend la forme d'une requête qui définit : les lieux de prise en charge et de livraison associés à la fenêtre de temps de service souhaitée. Nous définissons une solution comme un planning pour chaque véhicule qui répond aux demandes en satisfaisant leurs contraintes, tout en minimisant le temps d'attente des voyageurs et le coût de déplacement des véhicules. Les véhicules communiquent par diffusion via un réseau V2V, où la portée de communication est limitée. Chaque véhicule qui reçoit une nouvelle information la rediffuse et les véhicules sont ainsi connectés par transitivité dans les limites des rayons de communication. Dans ce contexte, un véhicule ne connaît pas les demandes en dehors de sa zone de communication. Cette connaissance évolue pendant le temps d'exécution, car le véhicule se déplace, reçoit de nouvelles demande des sources de demandes, rencontre d'autres véhicules et échange des messages entre eux.

Ce modèle AVFAP est défini comme suit :

$$
\begin{aligned}
AVFAP &:=< M, V, D, T > \\
M &:=< G, w > \\
V &:= \{v_1, v_2, \ldots, v_n\} \\
D &:= \{d_1, d_2, \ldots, d_m\} \\
T &:= \{t_0, t_1, \ldots, t_{end}\}
\end{aligned}
$$

Ici, M définit la carte des infrastructures urbaines (emplacements, routes et distances) sous la forme d'un graphique dirigé G associé à une fonction de valuation w ; l'offre est représentée par une flotte V de n véhicules autonomes ; D définit un ensemble dynamique de demandes de passagers qui apparaissent au moment de l'exécution ; et T définit l'horizon temporel dans lequel les véhicules doivent répondre aux demandes des passagers. Nous définissons le temps T comme un ensemble discret de $ticks$.

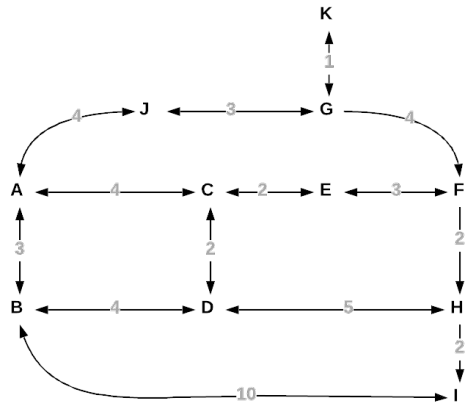

FIGURE 2 – Un exemple d'infrastructure urbaine

3.1 Propriétés de l'infrastructure routière

La carte des infrastructures routières est définie par un graphe dirigé pondéré M, comme le montre la Figure 2.

$$
\begin{aligned}
G &:=< N, E > \\
w &:= \{w_{e_1}, w_{e_2}, \ldots, w_{|E|}\}
\end{aligned}
$$

G est un graphe connecté dirigé où N est l'ensemble des nœuds, E est l'ensemble des arcs entre les nœuds. La fonction de valuation w associe chaque arc e à la valeur w_e selon une mesure de distance temporelle (par exemple, le temps de conduite moyen en minutes), qui sera utilisée pour calculer les coûts opérationnels des déplacements des véhicules.

3.2 Propriétés des véhicules

Chaque véhicule $v \in V$ est défini par

$$v :=< capa, cpd, range >$$

et ses propriétés dynamiques sont définies par

$$
\begin{aligned}
v\_location &: V \times T \to N \cup E \\
free\_seats &: V \times T \to \mathbb{N}^+
\end{aligned}
$$

où $capa$ définit le nombre total de sièges passagers dans le véhicule, cpd définit le coût du véhicule par unité de distance, et $range$ définit la portée de communication dans laquelle le véhicule peut communiquer avec d'autres entités. A tout moment $t \in T$ un véhicule $v \in V$ connaît sa situation courante : $v\_location$ définit l'emplacement de v qui pourrait être situé dans un nœud $n \in N$ ou se déplaçant sur un arc $e \in E$; et $free\_seats$ définit le nombre de places libres disponibles dans v au moment t.

3.3 Propriétés des demandes

Une demande $d \in D$ est définie comme

$$d :=< required, tw, pick\_up, drop\_off >$$

Pour une demande $d \in D$, $required$ est le nombre de sièges requis ; tw définit un intervalle de temps (t_{min}, t_{max}) dans

lequel l'événement de ramassage est considéré comme acceptable ; $pick\_up$ et $drop\_off$ sont respectivement l'origine et la destination de la demande. Comme pour les véhicules, au temps $t \in T$, nous considérerons qu'une demande $d \in D$ peut également être communiquée en utilisant le réseau V2V. La demande pourrait être émise par le client ou l'infrastructure (unité de bord de route).

4 Insertion basée sur des enchères

Dans notre modèle, nous utilisons une heuristique d'insertion comme celle décrite par Solomon [1987] pour adapter en continu les plannings locaux des véhicules. Le résultat de cet algorithme est un ensemble de demandes avec pour chacune l'horaire auquel un véhicule sera à la position de son origine. Chaque agent détermine ses horaires pour maximiser la valeur de la qualité de sa solution. Comme plusieurs véhicules peuvent être intéressés par une même demande, nous avons besoin d'un mécanisme de coordination pour résoudre ces conflits. Nous utiliserons pour cela un mécanisme d'enchères, qui est l'un des moyens efficaces et éprouvés pour résoudre de tels problèmes [Cramton et al., 2007].

4.1 Critère de l'enchère

Lorsqu'un véhicule v a connaissance d'une demande d, il la classe dans sa file d'attente selon la priorité qu'il lui a attribuée.

Au temps t, v choisit la première demande d_s dans la file d'attente, génère un ensemble d'alternatives, chacune étant un planning potentiel résultant de l'insertion de d_s dans une étape réalisable du planning actuel de v. Le coût opérationnel marginal de l'ajout de cette demande au planning est noté $cost$. Le choix avec le meilleur $cost$ est considéré pour diffuser une offre

$$Bid_v^d(t_{start}, cost)$$

avec t_{start} le moment de $pick\_up$ pour d_s.

Dans ce document, nous considérons le coût opérationnel d'un voyage comme la longueur totale de son trajet (somme des distances indiqués dans la Figure 2). Le coût marginal d'insertion est donc la différence de longueur de trajet entre le trajet initial et le nouveau trajet. Les offres restent disponibles pendant une période de temps spécifique t_{expire}. Ainsi, si le coût de l'offre de v est inférieur à toute autre offre reçue à $t + t_{expire}$ pour servir une demande d, il se considère comme gagnant de l'enchère, et met à jour son planning avec le nouveau chemin de l'offre.

Exemple 1 *Le scénario simple de la Figure 3 montre deux véhicules V_1 et V_2 situés en A et B, en état de repos avec des plannings vides au début. Au temps t_1, la première demande est annoncée $d_1 :< 1, (t_{10}, t_{20}), C, H >$. Les deux véhicules connaissent maintenant d_1. V_1 peut le servir en suivant le chemin $A \to C \to E \to F \to H$, donc V_1 place l'offre $Bid_{V_1}^{d_1}(t_{10}, 11)$. V_2 peut la servir via le chemin $B \to D \to C \to E \to F \to H$, donc émet l'offre $Bid_{V_2}^{d_1}(t_{10}, 13)$. V_1 est considéré comme un gagnant et ajoute d_1 à son planning, de sorte que le coût opérationnel global de la flotte est maintenant* 11.

4.2 Le besoin d'échange de demandes

Afin de pouvoir faire des offres efficaces pour de nouvelles demandes ou pour améliorer la solution, nous proposons que les véhicules échangent leurs demandes planifiées.

Exemple 2 *La Figure 4 montre une situation où l'utilisation de l'heuristique d'insertion fondée sur les enchères est très réactive mais ne garantit pas de bons plannings. Au moment t_2 où la nouvelle demande $d_2 :< 1, (t_{15}, t_{40}), J, K >$ arrive, les deux véhicules en prennent connaissance et placent leurs éventuelles offres. En l'absence de capacité d'échange, V_1 a encore d_1 dans son planning (avec un coût initial de* 11*), donc la meilleure offre qu'il peut placer est de servir les deux demandes (servir d_2 puis aller servir d_1) avec un coût marginal de* 14*. Alors que V_2 place l'offre gagnante $Bid_{V_2}^{d_2}(t_{15}, +11)$, il ajoute d_2 à son planning, et le coût global devient* 22*. Notez que dans ce cas, servir d_2 avec V_1 et laisser V_2 s'occuper de d_1 entraîne un gain de coût opérationnel global de* 21*, mais cette solution n'est jamais réalisée car d_1 est déjà programmé sur V_1.*

Pour parvenir à ce type de solution, nous devons permettre

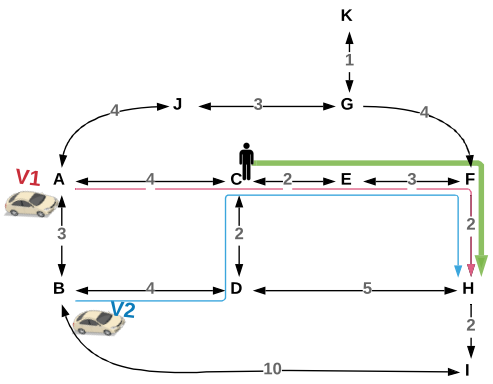

FIGURE 3 – V_1 remporte l'enchère pour servir d_1 de C à H

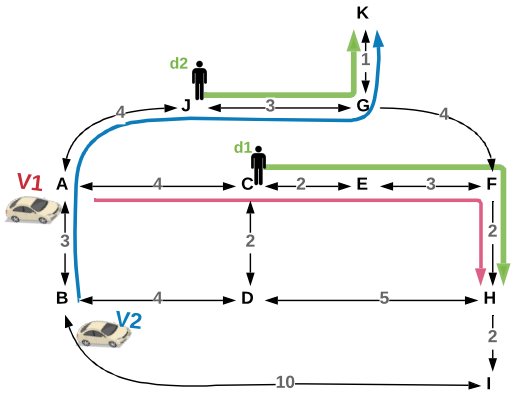

FIGURE 4 – Sans échange de la demande, V_2 gagne d_2 et V_1 garde d_1

aux véhicules d'échanger leurs demandes déjà programmées.

5 Améliorer la qualité des solutions

Dans ce qui suit nous proposons un protocole d'optimisation locale pour améliorer la qualité de la solution pour les cas comme celui illustré Figure 4 et l'exemple 3

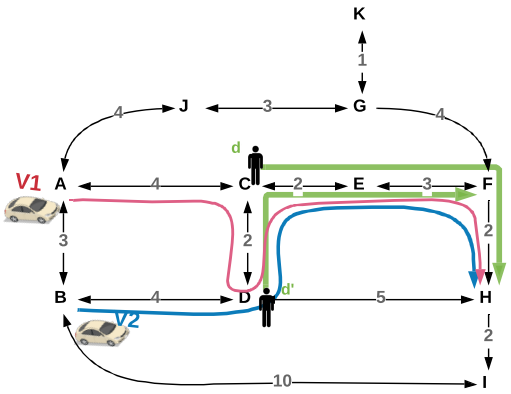

FIGURE 5 – Même s'il peut servir les deux demandes, V_2 ne peut soumettre une offre que pour une seule demande à la fois

5.1 Protocole d'optimisation « *Pull-demand* »

À l'instar de la stratégie *Rolling Horizon* de Agatz et al. [2011], nous proposons un protocole d'optimisation d'offres pour améliorer notre heuristique. Dans la stratégie *Rolling Horizon*, tous les plannings des véhicules sont considérés comme temporaires et disponibles pour être planifiés par n'importe quel véhicule, à moins qu'ils ne soient considérés comme des *demandes engagés* par des événements particuliers (par exemple, v a commencé à servir (se diriger vers) d, dont le temps restant pour le servir est inférieur au seuil de l'horizon). L'application de cette stratégie exige que tous les plannings des véhicules soient en mémoire partagée, de sorte que lorsqu'un véhicule v_i offre de servir une demande d, il sache s'il est programmé par un autre véhicule v_j, et donc s'il doit envoyer son coût d'offre à v_j. Ensuite, v_j calculera le gain (ou la perte) de coût opérationnel en abandonnant d et le coût pour v_i. S'il y a un gain, il accepte d'abandonner d et ensuite v_i met à jour son planning avec d, sinon l'offre est rejetée.

Dans notre proposition de protocole, nous ne faisons pas appel au concept de *demande engagée*, mais un véhicule ne peut faire des offres que pour des demandes qu'il peut satisfaire, de sorte que les demandes qui sont reprises ou qui n'ont pas assez de temps pour être reprogrammées sont automatiquement ignorées par l'agent. Une autre différence ici est que nous n'avons pas de mémoire partagée. Les agents échangent des informations sur le contexte de l'environnement et sur les demandes par des messages d'information. En plus de la proposition de Agatz et al. [2011], où l'optimisation est effectuée périodiquement à une fréquence prédéfinie, le protocole que nous proposons doit être exécuté en parallèle avec une stratégie fondée sur des enchères d'insertion, sur la base des informations partagées sur le contexte courant pour avoir une replanification rapide pour les demandes en continu. Le **Protocole 1** détaille cette stratégie.

Protocole 1 Protocole d'optimisation « *Pull-demand* »

Étape 1 Des nouvelles demandes entre dans le système par ordre d'annonce.

Étape 2 Chaque une est diffusée dans l'ensemble connecté auquel les sources appartiennent. Chaque agent de cet ensemble sélectionne ses demandes potentielles parmi toutes leurs demandes : les nouvelles demandes, les demandes planifiées et non planifiées qui n'ont pas encore atteint leur heure de départ prévue.

Étape 3 Les agents débutent l'enchère pour servir leurs demandes potentielles.

Étape 4 Chaque agent recherche parmi ses demandes planifiées celle à satisfaire au prochain tick, cette demande est appelée d_{next}. Si d_{next} existe, l'agent diffuse un message "$clear\_demand$" pour informer les autres agents de sa prise en charge de d_{next}. Chaque récepteur efface cette dernière de leurs ensembles de demandes potentielles et connues. Chaque agent efface toute autre demande qui atteint sa limite de temps.

Étape 5 Les demandes programmées et non programmées qui ont encore du temps restent diffusées par leurs sources (*Étape 2*). Cela permet une meilleure planification dans le prochain tick, étant donné que de nouvelles demandes peuvent être annoncées.

5.2 Discussion

Compte tenu du contexte décentralisé, l'utilisation de l'heuristique d'insertion est très efficace en termes de temps de réponse. La complexité temporelle de l'heuristique d'insertion de base pour le VRP est en $\mathcal{O}(n^3)$[Campbell et Savelsbergh, 2004]. Ce type d'heuristique est souvent utilisé pour résoudre des DARP, où les nouvelles demandes entrantes doivent être traitées en continu et intégrées dans les plannings évolutifs des véhicules. Dans l'exemple suivant, nous illustrons comment le protocole « *Pull-demand* » peut apporter une amélioration notable de la qualité de la solution.

Exemple 3 *Examinons un autre cas illustré dans la Figure 5, où V_1 possède d dans son planning, V_2 a un planning vide, et une nouvelle demande $d' :< 1, (t_{10}, t_{20}), D, F >$ est annoncée. Les offres des véhicules sont donc $Bid^{d'}_{V_1}(t_{12}, +4)$, et $Bid^{d'}_{V_2}(t_{10}, +11)$. Examinons le protocole mis en œuvre, étape par étape.*

Étape 1 d' entre dans le système à t_2 et les deux véhicules

en sont conscients, V_1 remporte l'enchère avec un coût global de 15

Étape 2 d *et* d' *sont maintenant dans l'ensemble de demandes connues par les deux véhicules,* V_2 *calcule les coûts pour servir* d *seule (*13*),* d' *seule (*9*) et les deux demandes ensemble ce qui fait* 13*. Il sélectionne donc les deux demandes comme ses demandes potentielles.* V_1 *n'a pas de demande potentielle car il a déjà les deux demandes dans son planning*

Étape 3 V_2 *place une offre* $Pull\_Bid_{V_2}^{d',d}((t_{10}, t_{12}), 13)$*. Pour* V_1 *le coût pour servir les deux demandes est de* 15 *donc il accepte* $Pull\_Bid_{V_2}^{d',d}$ *car cela engendre un gain de* 2*.*

Étape 4 Aucune des demandes n'atteint son temps de service prévu ou la limite supérieure de sa fenêtre temporelle.

Étape 5 Toutes les demandes connues restent diffusées et disponibles pour la prochaine amélioration potentielle.

V_2 gagne et la solution est améliorée avec un cycle d'optimisation supplémentaire.

6 Évaluation expérimentale

Dans cette section, nous évaluons expérimentalement la performance de notre approche « *Pull-demand* », en utilisant des données générées et des données cartographiques ouvertes.

6.1 Configuration expérimentale

La ville de Saint-Étienne a été choisie pour la simulation. La structure du graphique $G =< N, E >$ incluant les nœuds, les arêtes et un ensemble de sources d'émission de la demande $S \subset N$ est extraite d'OpenStreetMap (OSM [1]). Dans toutes les expérimentations, nous avons fixé le nombre de sources $|S| = 20$, ayant un ensemble $E_S \subset E$, tel que $|E_S| = 75$ arcs reliant les sources, chaque arc a un nombre de points qui varie en fonction de sa longueur et des informations extraites d'OSM. La distance entre deux points consécutifs est de 40 mètres. Nous avons utilisé un simulateur de transport en temps discret disponible dans la *Plateforme Territoire* [2] pour évaluer la stratégie proposée et l'analyser en termes de qualité de service et de gain.
Une flotte V de n véhicules est distribuée au hasard sur S au début de l'exécution. Chaque véhicule $v \in V$ se déplace d'un point à un autre sur le même arc au cours de chaque cycle de simulation. Dans notre test, nous considérons que les véhicules communiquent via V2V avec une portée de communication réaliste de 250 mètres, de sorte qu'un véhicule peut envoyer/recevoir des messages vers/depuis d'autres entités situées dans sa portée. Chaque véhicule v peut adopter deux comportements de déplacement distincts : soit *marauder* pour les demandes, soit *aller* vers une destination :

— `marauding` définit l'état d'un véhicule lorsqu'il n'a pas de demande à servir, c'est-à-dire qu'au début de la simulation, chaque véhicule maraude jusqu'à ce qu'il décide de servir une demande. Une fois qu'un passager est déposé, le véhicule repasse en `marauding`. Dans cet état, le véhicule se déplace aléatoirement dans son quartier afin de trouver des demandes à servir.

— `going_to` définit l'état d'un véhicule lorsqu'il a une destination spécifique, c'est-à-dire qu'une demande de desserte du véhicule est soit *going_to* lieu de prise en charge si la demande n'est pas encore prise en charge, soit *going_to* lieu de livraison sinon.

À chaque cycle de simulation, 0 ou 1 demande est générée de façon aléatoire uniforme. Pour chaque demande, les points d'origine et de destination sont générés de manière aléatoire et uniforme à partir de l'ensemble des sources. La fenêtre de temps pour les demandes est générée en utilisant deux paramètres constants l et u pour les limites inférieure et supérieure comme suit. $[tw_{min}, tw_{max}]$ est initialisée avec deux valeurs aléatoires uniformes où :

$$tw_{min} < tw_{max}$$

$$tw_{min} \geq t_{actuel} + l$$

$$tw_{max} \leq tw_{min} + u$$

Les critères d'évaluation de ces simulations sont principalement le nombre de demandes satisfaites comme mesure de la qualité de service (QoS), le bénéfice simulé de la solution comme mesure de la qualité des affaires (QoB). Le bénéfice est calculé en termes de différence entre le prix du voyage et le coût.

$$profit = total\_income - total\_cost$$

où

$$total\_income = \sum_{d \in D_s} P + p * distance(d)$$

$D_s \subseteq D$ est l'ensemble de toutes les demandes satisfaites, P est un prix fixe par demande, p est un facteur de tarification par unité de distance parcourue, $distance(d)$ est la distance totale de déplacement pour une demande d et

$$total\_cost = \sum_{v \in V} cpd(v) * total\_distance(v)$$

Dans nos tests, nous considérons que les véhicules sont identiques en ce qui concerne le coût de déplacement $cpd(\cdot)$ et le facteur de tarification p. Nous avons fixé à $P = 1.5$, $p = 2$ et $cpd(v) = 1$ pour tout v.

1. https://www.openstreetmap.org/
2. https://territoire.emse.fr/

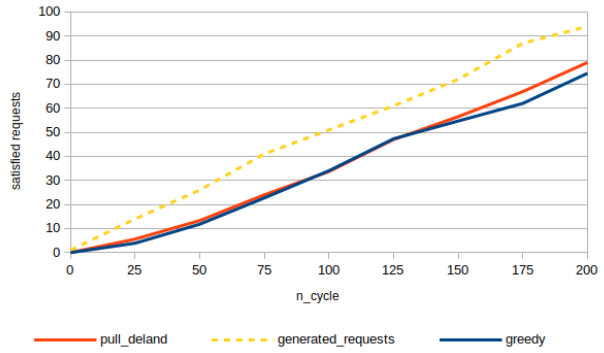

FIGURE 6 – QoS pour une flotte de 16 véhicules

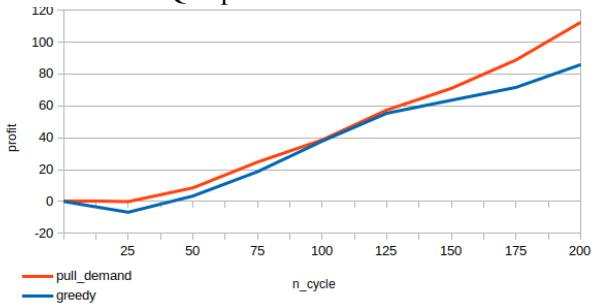

FIGURE 7 – QoB pour une flotte de 16 véhicules

6.2 Résultats expérimentaux

Pour évaluer la faisabilité de l'approche heuristique « *Pull-demand* », nous la comparons à une approche gloutonne (programmer une seule demande à l'avance) qui a été mentionnée par van Lon et al. [2012] comme la meilleure stratégie pour les réglages dynamiques, suite à une sélection génétique. Les deux approches comparées utilisent la même stratégie de sélection des demandes (fonction de priorité des demandes) au cours de chaque scénario. Les résultats présentés dans les Figures 6 et 7 concernent un scénario dans lequel les véhicules choisissent la demande la moins chère. La fonction de priorité retourne pour une demande d une valeur qui est inversement proportionnelle au coût de la demande pour v : $1/costDemand(d, v)$. Ainsi plus le coût est faible plus la priorité est élevée. $costDemand(d, v)$ est le coût marginal mentionné dans la section 4 ; il en va de même pour l'algorithme glouton, puisque la planification des véhicules ne peut contenir qu'une seule demande. Nous avons exécuté plusieurs instances de problèmes variant en fonction de la taille de la flotte n. Chaque instance de ces tests est exécutée 10 fois avec différentes graines aléatoires. Tout d'abord, nous évaluons avec une taille de flotte fixe $n = 16$ pour suivre la qualité de la solution dans le temps, puis avec une taille de flotte variable $n \in [4, 36]$ sur 200 cycles pour chaque scénario. Il est à noter qu'il est impossible d'atteindre un taux de qualité de service de 100% dans ces scénarios, car il y aura toujours des demandes qui seront générées jusqu'au dernier cycle, en supposant que l'exécution du scénario continue à les servir, alors que l'exécution s'arrête au dernier cycle, les laissant non servies. Bien que les valeurs de QoS de notre algorithme soient proches de

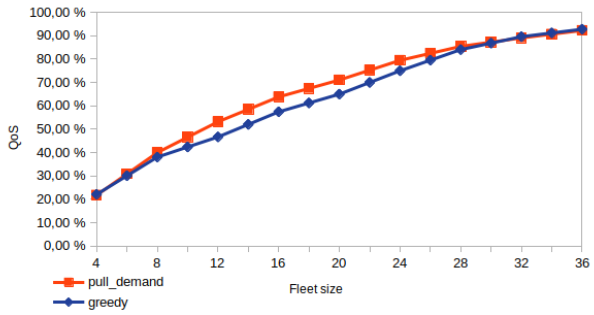

FIGURE 8 – QoS avec des flottes de tailles croissantes

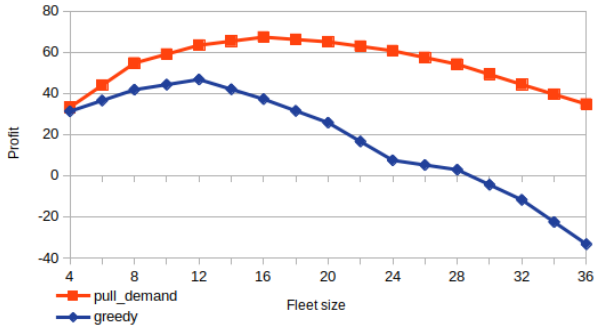

FIGURE 9 – QoB avec des flottes de tailles croissantes

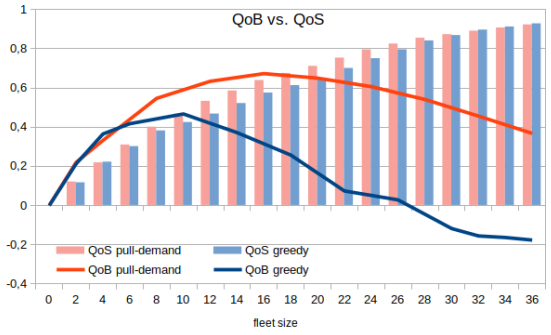

FIGURE 10 – Évolution de QoS vs. QoB

celles générées par l'approche gloutonne, elles restent légèrement supérieures à celles-ci dans la plupart des cas, comme le montre la Figure 8. En partant d'une petite flotte de 4 véhicules, l'heuristique « *Pull-demand* » permet d'obtenir des solutions avec les mêmes valeurs de QoS et de QoB que celles de l'algorithme glouton. En augmentant la taille de la flotte, plus de demandes peuvent être satisfaites, ce qui augmente la valeur de QoS et de QoB. Nous constatons que l'augmentation de ces valeurs pour l'heuristique est plus importante que celle de l'approche gloutonne. Cela peut s'expliquer par le fait que la planification des demandes améliore le profit du système dans un avenir prévisible, car elle réduit le processus de déplacement sans destination spécifique jusqu'à ce qu'une nouvelle demande soit choisie, ce qui est le comportement des véhicules dans l'algorithme glouton.

La valeur de la qualité de service continue de croître lorsque

la taille de la flotte augmente, jusqu'à atteindre un seuil n_{QoS} pour lequel l'ajout de nouveaux véhicules devient inutile, car toutes les demandes reçues dans le système maintenant ou à l'avenir (sauf celles qui sont reçues dans les derniers moments d'exécution) peuvent être servies avec la taille actuelle de la flotte. Il en va de même pour la QoB, mais la différence ici est que les véhicules ajoutés entraîneront des dépenses opérationnelles supplémentaires, ce qui entraîne une perte de valeur du bénéfice après avoir atteint son seuil de croissance n_{QoB}. Habituellement, n_{QoB} est inférieur à n_{QoS}, et nous pouvons observer un compromis entre l'amélioration de la QoS ou de la QoB.

D'après la Figure 9, nous constatons que la flotte gloutonne dispose de $n_{QoB}^{\text{greedy}} = 12$ alors que sa QoS n'est que d'environ 60%, tandis que $n_{QoB}^{\text{pull-demand}} = 16$ avec une QoS d'environ 70%. Notez que $n_{QoB}^{\text{gloutonne}} < n_{QoB}^{\text{pull-demand}}$ et même si elle diminue ensuite, la valeur de $n_{QoB}^{\text{pull-demand}}$ reste supérieure à $n_{QoB}^{\text{gloutonne}}$. Cela donne un plus large éventail d'options pour combiner les améliorations de la solution fournie (QoS et QoB). Les résultats ci-dessus montrent que dans tous les cas, la programmation de plusieurs demandes à l'avance avec le protocole d'optimisation « *Pull-demand* » donne de meilleurs résultats que la programmation d'une seule demande, tout en nécessitant un temps de calcul réduit, et aucune information globale.

7 Conclusion

Nous avons proposé dans cet article un protocole décentralisé pour l'échange de demandes, s'appuyant sur une heuristique d'insertion, pour allouer des demandes aux véhicules dans le contexte du transport à la demande dynamique. Nous montrons au travers des exemples que le protocole de demande d'échange de demandes peut être une amélioration prometteuse de la qualité des solutions. Afin d'évaluer la faisabilité du protocole proposé, nous avons évalué les résultats de notre technique sur des données synthétiques pour les taxis opérant dans la ville de Saint-Étienne, et avons montré qu'elle surpasse une approche gloutonne classique. Dans les travaux futurs, nous prévoyons d'évaluer l'efficacité, la performance, la robustesse et l'optimalité de cette heuristique par rapport à différentes approches, en simulant différents paramètres sur la distribution de l'information, les critères de décision et différents niveaux de dynamique du problème.

Références

N. A. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang. Dynamic ride-sharing : A simulation study in metro atlanta. *Transportation Research Part B : Methodological*, 45(9) :1450 – 1464, 2011.

A. M. Campbell and M. Savelsbergh. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation science*, 38(3) :369–378, 2004.

P. Cramton, Y. Shoham, and R. Steinberg. An overview of combinatorial auctions. *ACM SIGecom Exchanges*, 7 (1) :3–14, Dec. 2007.

P. Danassis, A. Filos-Ratsikas, and B. Faltings. Anytime Heuristic for Weighted Matching Through Altruism-Inspired Behavior. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 215–222, Macao, China, Aug. 2019.

K. C. Dey, A. Rayamajhi, M. Chowdhury, P. Bhavsar, and J. Martin. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network – Performance evaluation. *Transportation Research Part C : Emerging Technologies*, 68 : 168–184, July 2016.

M. Egan and M. Jakob. Market Mechanism Design for Profitable On-Demand Transport Services. *arXiv :1501.01582 [cs]*, Jan. 2015.

M. El Falou, M. Itmi, S. El Falou, and A. Cardon. On demand transport system's approach as a multi-agent planning problem. In *2014 International Conference on Advanced Logistics and Transport (ICALT)*, pages 53–58. IEEE, 2014.

A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev. Multi-Agent Real Time Scheduling System for Taxi Companies. *AAMAS*, page 8, 2009.

J. M. S. Grau and M. A. E. Romeu. Agent Based Modelling for Simulating Taxi Services. *Procedia Computer Science*, 52 :902–907, 2015.

J. M. S. Grau, M. A. E. Romeu, E. Mitsakis, and I. Stamos. Agent based modeling for simulation of taxi services. *Journal of Traffic and Logistics Engineering*, 1(2) :159–163, 2013.

G. Picard, F. Balbo, and O. Boissier. Approches multiagents pour l'allocation de courses à une flotte de taxis autonomes. In *RIA2018*, number 2 in Revue d'intelligence artificielle, pages 223–247, 2018.

W. Shen and C. Lopes. Managing Autonomous Mobility on Demand Systems for Better Passenger Experience. In *PRIMA 2015 : Principles and Practice of Multi-Agent Systems*, volume 9387, pages 20–35, Cham, 2015. Springer International Publishing.

M. M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2) :254–265, 1987.

R. R. van Lon, T. Holvoet, G. Vanden Berghe, T. Wenseleers, and J. Branke. Evolutionary synthesis of multi-agent systems for dynamic dial-a-ride problems. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*, page 331, Philadelphia, Pennsylvania, USA, 2012. ACM Press.