Corwin Fèvre, Philippe Mathieu, Hayfa Zgaya-Biau, Slim Hammadi

Covoiturage dynamique multi-saut avec modélisation des préférences utilisateur

Volume 5, nº 4 (2024), p. 37-61.

https://doi.org/10.5802/roia.86

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Volume 5, n° 4, 2024, 37-61

Covoiturage dynamique multi-saut avec modélisation des préférences utilisateur

Corwin Fèvre[a], Philippe Mathieu[a], Hayfa Zgaya-Biau[a], Slim Hammadi[a]

[a] Univ. Lille, CNRS, Centrale Lille, UMR 9189, CRIStAL, F-59000 Lille, France
E-mail : corwin.fevre@online.fr, philippe.mathieu@univ-lille,
hayfa.zgaya-biau@univ-lille.fr, slim.hammadi@centralelille.fr.

Résumé. — Dans cet article, nous proposons une approche multi-agent pour résoudre le problème du covoiturage dynamique multi-saut. Dans notre système, les passagers et les conducteurs sont représentés comme des agents autonomes et rationnels en perpétuelle interaction pour satisfaire leurs propres objectifs comme leur temps d'attente ou leur temps de trajet par exemple. Dans la solution proposée, les agents conducteurs et passagers ont une perception modélisée dynamiquement en utilisant des R-Arbres. Nous modélisons leurs préférences en matière de détour et de trajet et montrons l'impact de celles-ci sur la résolution d'une instance de covoiturage dynamique. Les résultats présentés montrent que notre système permet de traiter dynamiquement des requêtes complexes de passagers tout en minimisant l'impact du partage de trajet pour les conducteurs, et ce, pour un large spectre de préférences et de comportements.

Mots-clés. — Covoiturage, Simulation, Optimisation, Agents.

1. Introduction

Le covoiturage est devenu un moyen de transport à part entière, que ce soit pour les trajets quotidiens ou pour des voyages plus longs. Outre son intérêt écologique, il permet aux conducteurs de réduire les coûts liés à leurs déplacements tout en offrant la possibilité aux passagers de voyager avec plus de flexibilité que les moyens de transport conventionnels. Il en résulte une utilisation accrue des véhicules déjà déployés sur la route, une réduction de la congestion du trafic et donc une diminution des émissions polluantes.

Nous proposons dans cet article d'étudier le covoiturage dynamique entre particuliers. Un système de covoiturage dynamique prend en compte l'état actuel de l'environnement des utilisateurs pour effectuer des associations de covoiturage. Un tel système collecte en temps réel différents flux d'information dans l'environnement comme les itinéraires et temps de trajets des conducteurs en cours de route ainsi que les requêtes des passagers. Le caractère dynamique réside alors dans le flux continu de conducteurs et de passagers entrant, évoluant et sortant du système. Cette forme de covoiturage est majoritairement étudiée à l'échelle de grandes métropoles [11, 30] et dans le cadre de trajets courts et spontanés [21, 31].

La flexibilité d'un système de covoiturage dynamique peut être modélisée et quantifiée par deux opérations. D'une part, les conducteurs peuvent choisir de faire des détours pour prendre ou déposer des passagers, et ainsi élargir l'offre de transport du système de covoiturage. D'autre part, les passagers peuvent choisir de passer d'un véhicule à l'autre, et donc d'effectuer des transferts, afin d'arriver à leur destination en plusieurs étapes, si cela est plus pratique pour eux. Un exemple de ces opérations est illustré dans la figure 1.1.

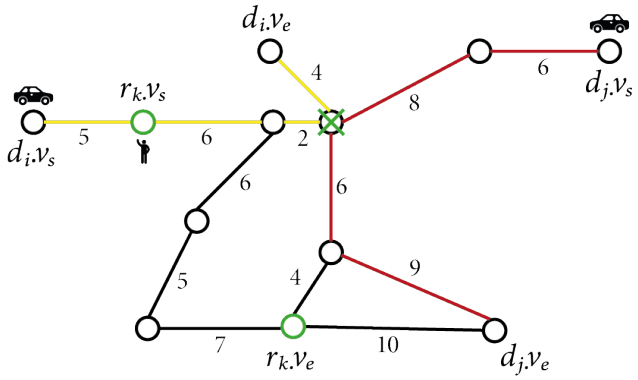

FIGURE 1.1. Un exemple de covoiturage multi-saut

Dans cette figure, deux conducteurs d_i et d_j suivent leur itinéraire, respectivement en jaune et rouge, reliant leur nœud de départ v_s et leur nœud d'arrivée v_e. Un passager r_k attend sur son nœud de départ $r_k.v_s$ et veut rejoindre son nœud d'arrivée $r_k.v_e$, tous deux entourés de vert. Les trajectoires des conducteurs se croisent sur le nœud avec une croix verte, un transfert est alors réalisable. Le conducteur d_i peut partager son trajet avec le passager r_k puis, une fois arrivé au nœud de transfert, le passager peut effectuer un transfert avec le conducteur d_j. Le conducteur d_j doit cependant faire un détour pour déposer le passager, ce qui rajoute 5 unités de temps à son temps de trajet.

Le défi d'un tel cadre est d'équilibrer l'impact du covoiturage sur l'offre et la demande. Cet équilibre implique de nombreux paramètres tels que le ratio conducteurs-passagers ou les détours acceptables par les utilisateurs en termes de temps et de distance. Le problème est alors d'associer au mieux les passagers et conducteurs en exploitant leur marge de manœuvre afin de satisfaire les deux parties.

Un système de covoiturage peut être naturellement représenté par un système multi-agent, les conducteurs et les passagers étant des entités autonomes en interaction. C'est pourquoi nous proposons dans cet article un système multi-agent permettant de décrire des comportements spécifiques pour chaque utilisateur, garantissant le caractère hétérogène de notre simulation. La limite de perception de ces agents est définie dynamiquement par leurs préférences et contraintes spatio-temporelles. Ils disposent ainsi d'une rationalité limitée. L'intérêt d'une telle approche réside dans sa capacité à simuler fidèlement des micro-phénomènes et de garantir une certaine indépendance et

autonomie des acteurs du système, contrairement à des approches plus macros visant à simuler le système dans sa globalité et les individus comme une population uniforme.

Pour traiter l'important flux d'informations généré par un tel système, nous utilisons une structure d'indexation spatiale nommée R-Arbre [23]. Cette structure, issue de la recherche dans le domaine GIS (geographical information system), permet de stocker efficacement et dynamiquement des données multidimensionnelles ainsi que d'effectuer des requêtes de superposition sur des points et des zones de l'espace. Les algorithmes associés aux R-Arbres permettent un zonage dynamique de l'information avec une répartition intelligente de la charge de l'exploration spatiale, tandis que la majorité de la littérature utilise un découpage statique en zone de même taille (rectangles, pentagones...) impliquant parfois un phénomène de « point chaud » dans l'exploration spatiale.

Les contributions de cet article peuvent être résumées de la façon suivante :

- Nous proposons une approche centrée individu pour le problème du covoiturage en détaillant un système multi-agent pour simuler différents comportements de conducteurs et de passagers ainsi que leurs interactions.
- Nous étendons le champ d'application de la technique d'indexation spatiale R-Arbre au covoiturage entre particuliers afin de modéliser et d'identifier efficacement et dynamiquement les opportunités de covoiturage et de transfert.
- Nous considérons ce problème de covoiturage d'une manière multi-objectifs et proposons une fonction d'évaluation agrégée pour permettre aux usagers de prendre une décision dans l'espace des solutions.
- Nous expérimentons notre approche sur une carte réelle et en simulant des comportements proches de la réalité. Cette étude montre l'impact du profil des préférences des passagers et des conducteurs sur la résolution de différentes instances, à la fois au niveau individuel et au niveau global.

Après un état de l'art sur le problème du covoiturage et ses approches d'optimisation et de simulation dans la section 2, nous définissons notre problème de covoiturage en section 3. Nous proposons en section 4 une solution basée sur un système multi-agent dont le comportement des agents est défini par divers algorithmes. En section 5 nous étudions différents comportements typiquement observables dans des scénarios réels de covoiturage. Nous les confrontons à notre modèle et étudions les résultats obtenus sur des cartes réelles. Nous concluons sur les limites et perspectives de notre approche en section 6.

2. Travaux connexes

À l'occasion d'un covoiturage, un passager peut être acheminé par un seul et unique conducteur, c'est ce que l'on nomme **le covoiturage simple saut** (ou *single-hop* en anglais). Le problème du covoiturage simple-saut consiste à identifier le meilleur conducteur pour acheminer un passager entre deux positions. C'est la première forme de covoiturage étudiée dans la littérature [2, 11, 19] de par ses similarités avec les problèmes d'allocation optimale de ressources.

Il est aussi possible pour un passager d'utiliser plusieurs conducteurs, et donc d'effectuer des transferts afin d'arriver à sa destination. Cette variante s'appelle **le covoiturage multi-saut** (ou *multi-hop* en anglais) ou encore le covoiturage avec transferts [9]. Cette méthode permet d'améliorer les résultats obtenus avec le covoiturage simple-saut en augmentant les possibilités de covoiturage [17, 31]. Le covoiturage multi-saut implique une complexité accrue, car il faut cette fois identifier la meilleure combinaison de conducteurs ainsi que le lieu de transfert optimal pour tous les acteurs du covoiturage. Le problème du covoiturage multi-saut a été associé à de nombreux problèmes NP-Durs et NP-Complets de la littérature [8, 21].

Pour gérer cette complexité, l'espace de recherche doit être organisé et exploré de manière optimale. De nombreux travaux de recherche hiérarchisent l'espace [24] ou le divisent en zones pour en réduire la complexité [22, 30]. Ces divisions et réductions de l'espace sont cependant la plupart du temps effectuées de manière empirique et non dynamique forçant une approximation importante de l'espace et une faible répartition de la charge de la recherche spatiale. Dans ce contexte, une structure d'indexation spatiale connue sous le nom de R-Arbre (*R-Tree* [15]) permet d'indexer dynamiquement les objets dans un arbre et d'optimiser les requêtes de superposition de zones ou de nœuds de l'espace [23].

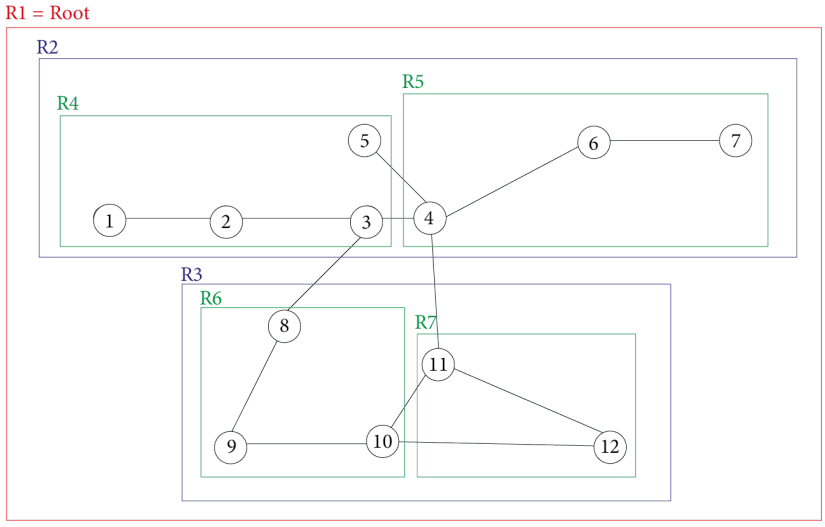

FIGURE 2.1. Exemple d'indexation R-Arbre sur le graphe de la figure 1.1. Les nœuds du graphe de routes numérotés de 1 à 10 sont indexés hiérarchiquement par des rectangles à limite minimum numérotés de R1 à R7.

Dans cet arbre, les nœuds sont des pages indexant hiérarchiquement d'autres pages ou objets, les feuilles sont des objets sous forme de rectangle à limite minimum (*MBR : minimum bounding rectangle* en anglais) ou de points, comme on peut le voir sur la figure 2.1. Ces objets sont identifiés par un identifiant unique, des coordonnées de

latitudes et de longitudes minimales et maximales et référence, éventuellement, un objet (ex. : un conducteur, un nœud du graphe d'itinéraire...).

L'intérêt d'une telle structure est d'une part d'être efficace dans un contexte dynamique. L'ajout, la suppression et la modification d'informations se font en parcourant l'arbre et en modifiant la disposition et le nombre de pages de manière optimisée à l'aide d'heuristiques (ex. : ajout à la page nécessitant le moins d'agrandissement). L'autre avantage des R-Arbres est la maîtrise de la complexité liée à la recherche d'objets présents dans une zone ou englobant un point ainsi qu'aux requêtes sur les plus proches voisins. Les algorithmes descendent récursivement dans l'arbre, en écartant les pages qui ne concernent pas la requête (en dehors de la zone de recherche...) afin d'éviter les opérations inutiles. Un exemple d'arbre de ce type est illustré à la figure 2.2. Chaque feuille est ensuite testée en fonction de la requête et renvoyée si les conditions sont remplies.

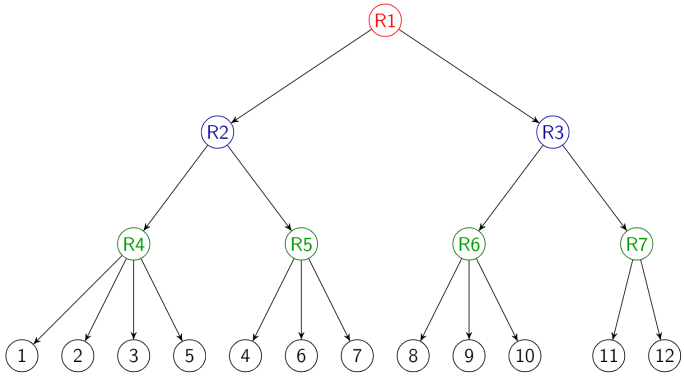

FIGURE 2.2. Arbre résultant de l'indexation R-Tree présentée dans la figure 2.1. Les trois premiers niveaux de l'arbre sont des nœuds correspondant aux rectangles de délimitation indexés hiérarchiquement. Le dernier niveau comprend les feuilles correspondant aux nœuds du graphe de route.

Cette structure a notamment été utilisée pour le partage de taxi et s'est révélée être très efficace [31]. Dans nos travaux, nous proposons d'étendre le champ d'utilisation des R-Abres au problème du covoiturage entre particuliers.

Il existe de nombreuses façons de modéliser le problème, notamment en ce qui concerne la granularité de la population. D'une part, l'approche centrée-groupe vise à optimiser des objectifs communs à l'ensemble de la population et donc à privilégier le système par rapport à l'individu [2, 11]. Cette méthodologie permet une simulation moins coûteuse en ressources, mais s'avère éloignée de la réalité vis-à-vis du caractère hétérogène de la population. En revanche, l'approche centrée-individu cherche à résoudre la demande de chaque utilisateur, au cas par cas, en tenant compte de ses préférences pour définir les objectifs d'optimisation. Ainsi, chaque individu a son propre comportement et sa propre perception. Dans ce contexte, l'approche multi-agent [13] permet de modéliser un tel système en garantissant l'autonomie de chaque utilisateur et en permettant à la simulation d'être plus proche de la réalité [4, 10, 18, 19, 25].

Le problème du covoiturage a d'abord été étudié de manière mono-objectif avec des approches de programmation par contraintes [2, 11]. Des problèmes de covoiturage multi-objectif ont ensuite été étudiés et résolus à l'aide d'heuristiques évolutionnaires et génétiques [16, 17]. Ces méthodes permettent une optimisation plus rapide - bien qu'approximative - et donc plus applicable dans le monde réel. Avec l'avènement du covoiturage multi-saut et afin d'obtenir une solution exacte et multi-objectif, la recherche s'oriente vers une approche du problème de covoiturage via le problème du/des plus proches voisins d'un groupe (GNN/ANN) [27]. En effet, puisque le multi-saut implique la recherche du nœud de transfert optimal, des heuristiques basées sur des algorithmes de recherche de plus proches voisins ont été développées [32] et ont été améliorées pour être plus efficaces dans un contexte réel [7, 31]. C'est dans cette perspective que nous proposons d'étendre l'algorithme de référence IER (Incremental Euclidean Restriction) [32] pour évaluer conjointement les solutions à sauts multiples et à sauts uniques de manière multi-objectif.

3. Formulation du problème

3.1. Spécification du réseau routier et des agents du système

Le réseau routier de notre système est modélisé par un graphe orienté et connexe $G = \langle V, E \rangle$ dans lequel V est l'ensemble des nœuds du graphe et E est l'ensemble des arêtes liant ces nœuds. Les arêtes représentent des routes et les nœuds représentent des intersections de routes ou des impasses. Un nœud $v_i = (x, y)$ est référencé spatialement par un couple de coordonnées : une longitude x et une latitude y. Différents poids peuvent être attribués aux arêtes désignant le coût du trajet entre deux nœuds, comme le temps de trajet par exemple.

Nous utilisons une simulation en temps discret, le temps évolue à intervalles constants et le pas de simulation courant est dénoté par la variable $time$. À chaque pas de simulation, tous les passagers et conducteurs présents dans le système peuvent effectuer une action en fonction de leur comportement.

Un agent utilisateur du système $u_i = \langle v_s, v_l, v_e, \det_{max} \rangle$, avec $u_i \in U$ représentant un passager ou un conducteur dans le système de covoiturage, est initialisé par quatre éléments : son nœud de départ v_s, son nœud de localisation actuel v_l, son nœud d'arrivée v_e et son facteur de détour $\det_{max}$. Le facteur de détour quantifie la volonté de détour d'un agent utilisateur. Ainsi, s'il vaut 0, l'utilisateur ne souhaite pas faire de détour, et s'il vaut 1, l'utilisateur est prêt à doubler son trajet initial pour effectuer des détours. Nous faisons varier ce facteur entre 0 et 1 dans nos expériences. Cet élément permet de définir les limites initiales d'action et de perception d'un agent.

Un agent conducteur $d_i = \langle v_s, v_l, v_e, \det_{max}, c_{max} \rangle$, avec $d_i \in D$, est initialisé par cinq éléments. Un conducteur étant un utilisateur, il hérite des quatre premiers paramètres définis précédemment. Le dernier paramètre c_{max} définit le nombre maximal de sièges disponibles dans le véhicule du conducteur. Un conducteur peut effectuer un nombre infini de détours et d'arrêts tant qu'il respecte son objectif d'atteindre sa destination avant son heure d'arrivée la plus tardive. Il suit donc son trajet de nœud en

nœud, en prenant systématiquement le chemin le plus court. S'il doit faire un détour pour prendre et déposer un passager, il doit être capable de déterminer si cette action est possible vis-à-vis de son heure d'arrivée la plus tardive. Pour ce faire, cet agent maintient un ordonnanceur de voyage où il stocke différentes informations telles que ses prochains arrêts *stops* (nœuds du graphe de route), les temps de trajets minimum *arr* et maximum *ddl*, les marges de détours *slk* ou les capacités restantes *c* (nombre de sièges disponibles) entre ses arrêts. La modélisation exacte de cet ordonnanceur est détaillée dans nos précédents travaux [14].

Un agent passager $r_i = \langle v_s, v_l, v_e, \det_{max}, wt_{max}, pref \rangle$, avec $r_i \in R$, est initialisé par six éléments. Un passager étant un utilisateur du système, il hérite des quatre premiers paramètres définis précédemment. Le paramètre wt_{max} correspond au temps d'attente acceptable au maximum par le passager pour la totalité du trajet. S'il est dépassé, l'agent passager s'impatiente et disparaît du système, ce qui affecte le taux de service global du système de covoiturage. La perception p d'un agent passager représente la zone des conducteurs accessibles. Elle permet d'exclure les conducteurs trop éloignés pour le prendre en charge. Le paramètre *pref* représente le vecteur de préférence du passager, c'est-à-dire son profil/comportement. Chaque élément $pref_i \in [0, \alpha]$ et la somme des éléments de *pref* doit être égale à α : $sum(pref_i, \ldots, pref_k) = \alpha$. On peut alors, par exemple, fixer α à 10 et faire varier la pondération des préférences d'un agent passager entre 0 et 10. De cette manière, nous pouvons attribuer un poids à chaque objectif contenu dans cette liste de préférences et détailler différents profils de passagers. Ceci implique que, si $pref_i > pref_j$, alors le passager accorde plus d'importance à la préférence $pref_i$ qu'à la préférence $pref_j$ lors de l'optimisation de son trajet. Un passager a donc pour objectif d'atteindre sa destination avant son heure d'arrivée la plus tardive tout en sélectionnant le ou les meilleurs covoiturages en fonction de ses préférences.

Un agent service de transport $tsa_i = \langle RT\_p \rangle$ est responsable des échanges entre les agents utilisateurs. Il remplit deux fonctions. La première consiste à stocker et à mettre à jour toutes les perceptions des agents conducteurs, à leur demande, lorsqu'un évènement entraîne une modification de leur perception. La seconde fonction est de répondre aux requêtes spatiales des agents passagers du système qui souhaite identifier des candidats au covoiturage. L'agent service de transport est donc implémenté comme un « tableau noir » afin de mettre à jour et de transmettre les informations des agents du système lorsque cela est demandé.

L'ensemble des interactions entre ces agents est résumé dans la figure 3.1.

3.2. Contraintes et Formulations

Nous limitons notre étude au covoiturage à un et deux sauts, c'est-à-dire au covoiturage limité à deux conducteurs pour un transfert. En effet, plusieurs articles comme [24] ont montré que l'augmentation du nombre de transferts possibles améliore à peine les performances d'un système de covoiturage tandis que la complexité

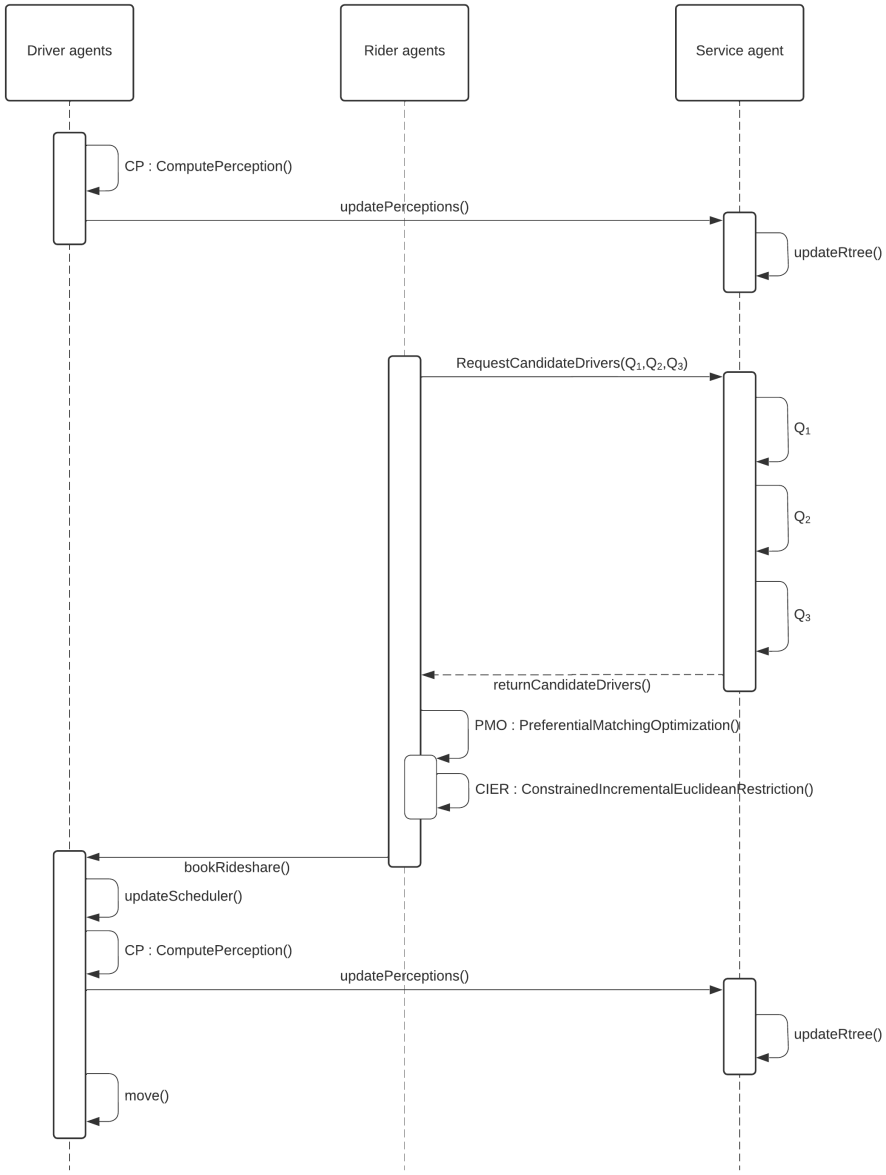

FIGURE 3.1. Le diagramme de séquence des agents du système. Les agents sont représentés par des lignes de vie. Les différentes actions et communications entre agents sont ordonnées chronologiquement de haut en bas. Les phases d'activation des agents sont représentées par les plus petits rectangles et les flèches représentent les appels de fonction et les communications interagents.

du problème explose de façon exponentielle. Dans cet article, chaque requête de covoiturage est associée à un seul et unique passager, c'est-à-dire que nous considérons qu'un passager n'a pas d'accompagnant quand il effectue une requête.

Pour qu'un covoiturage entre un passager et un conducteur soit réalisable, nous définissons plusieurs contraintes spatio-temporelles :

- **Contrainte temporelle :** Les conducteurs et les passagers doivent arriver à leur destination avant leur heure d'arrivée souhaitée. Un passager doit être pris en charge avant que son temps d'attente maximum ne soit atteint.
- **Contrainte de capacité :** La capacité courante d'un conducteur ne doit jamais dépasser la capacité maximale de son véhicule.
- **Contrainte de l'ordre des arrêts :** Un conducteur partageant son itinéraire avec un passager doit d'abord passer par le nœud de prise en charge du passager avant de visiter le nœud de dépôt de ce dernier.

Une association (*« a match »* en anglais) $m \in r_i.M$ est définie par plusieurs variables. Un ou deux conducteurs $d_j, d_k \in D$ (simple-saut : $j = k$, multi-saut : $j \neq k$) ; un nœud de transfert *v_tsf* s'il y a un transfert ; et enfin quatre critères : un nombre de transferts *nb_tsf* (0 ou 1 dans cette étude), un temps d'attente total *total_wt*, un temps d'arrivée *arr_time*, et une distance additionnelle de détour *add_dist*. Chaque passager a pour but de choisir l'association la plus optimale dans M selon ses préférences *pref*. Le problème étudié est donc multi-objectif en impliquant la minimisation simultanée de ces quatre critères :

- $total\_wt = start\_wt + tsf\_wt$: le temps d'attente total du passager correspond à la somme de son temps d'attente *start_wt* au nœud d'origine et au nœud de transfert *tsf_wt*. Cette valeur doit être inférieure au temps d'attente maximal, c'est-à-dire que pour un agent passager r_i : $r_i.total\_wt \leqslant r_i.wt_{max}$.
- $arr\_time = t_s + total\_wt + travel\_time$: l'heure d'arrivée de l'agent passager. Il correspond à la somme de son heure d'apparition t_s, de son temps d'attente *total_wt* et de son temps de trajet *travel_time*.
- $add\_dist = \sum_{i=1}^{nb\_d} (d_i.ridesharing\_dist - d_i.initial\_dist)$: la somme des distances additionnelles de détour des conducteurs impliqués dans le covoiturage du passager. La somme est effectuée pour le nombre de conducteurs *nb_d* impliqué dans le covoiturage. Une distance additionnelle de détour est calculée en effectuant la différence entre la distance de trajet du conducteur en covoiturant *ridesharing_dist*, et donc avec les éventuels détours, et sa distance de trajet initiale *initial_dist*. On ne considère alors pas tout le trajet du conducteur (car il pourrait y avoir d'autre covoiturage par la suite), mais uniquement le tronçon de covoiturage avec le passager.
- $nb\_tsf = nb\_d - 1$: le nombre total de transferts d'un agent passager pour atteindre sa destination. Il correspond au nombre de conducteurs impliqués dans le covoiturage *nb_d*, soustrait de 1. En effet, un covoiturage avec 1 seul conducteur n'implique pas de transfert (0) alors qu'un covoiturage avec 2 conducteurs implique 1 transfert.

La complexité de notre problème peut être résumée par : quel est le nombre de combinaisons de covoiturage possible dans le système ? Avec une approche individu-centrée, cette complexité peut être formalisée comme suit :

$$O(R * H^{D*N}) \tag{3.1}$$

avec H : le nombre de sauts possibles ; D : le nombre d'agents conducteurs ; R : le nombre d'agents passagers ; N : le nombre de nœuds du graphe de route. Ce problème est ainsi associé aux problèmes de classe de complexité exponentielle : $O(2^n)$.

4. Le système multi-agent proposé

À chaque étape de la simulation, le tour d'un agent utilisateur est divisé en deux phases : la phase de mise à jour de sa perception de l'environnement et la phase de prise de décision.

4.1. La mise à jour de la perception des agents

La perception d'un agent utilisateur représente ce qu'il est capable d'utiliser dans l'environnement pour atteindre son objectif. Dans nos travaux, cette perception est dynamique et doit être mise à jour régulièrement. Déterminer la perception exacte d'un agent utilisateur u_i est une tâche très lourde en termes de calcul. Il faudrait vérifier, pour chaque nœud v_j du graphe, si la somme du temps nécessaire pour aller de la position actuelle de l'agent au nœud v_j et du temps nécessaire pour aller de ce nœud v_j à la destination de l'agent est inférieure ou égale à l'heure d'arrivée la plus tardive t_e. Soit, $v_j \in u_i.p$ seulement si $SPt(G, u_i.v_l, v_j, u_i.w) + SPt(G, v_j, u_i.v_e, u_i.w) \leqslant u_i.t_e$. Nous devrions donc effectuer une recherche en largeur (*breadth-first* en anglais) et de nombreux calculs de plus courts chemins pour réaliser une telle tâche (un parcours par nœud et donc une complexité induite de $O(n^2)$). Il est difficilement concevable d'appliquer cette méthode sur une instance de problème de taille importante. Nous avons donc choisi d'approximer dans un premier temps la perception d'un agent en utilisant l'algorithme 1 nommé CP pour *Compute Perception*, de complexité $O(n \log(n))$.

Pour calculer la perception $p_{j,k}$ d'un agent u_i entre deux nœuds v_j et v_k du système, on détermine d'abord le plus court chemin entre ces deux nœuds (L1). On parcourt les nœuds de ce plus court chemin en cherchant les longitudes et latitudes minimales et maximales (L3-8). Ces limites nous permettent de créer le rectangle englobant minimal *MBR* (pour *« minimum bounding rectangle »*, en anglais) du plus court chemin de l'agent comme on peut le voir sur la figure 4.1.

Nous effectuons ensuite une extension du *MBR* avec le temps de marge de trajet de l'agent. Nous rappelons que cette marge est nommée *slk* et résulte des préférences de l'agent et de son état courant (détours effectués, temps restant...). Pour effectuer cette extension, nous récupérons la vitesse et la marge de temps de l'agent pour calculer la marge en termes de distance. Nous convertissons cette distance en degrés et étendons le *MBR* en conséquence (L9-18) afin d'établir le rectangle englobant étendu du trajet de l'agent, nommé *BBOX* (pour *« bounding box »*, en anglais). Enfin, nous restreignons

Algorithm 1 Compute Perception BBOX from slack time (CP)

Input : G = road infrastructure graph, *origin* = origin node, *dest* = destination node, *slk* = slack time
Output : *BBOX* = perception of the agent

1: $sp = SP(G, origin, dest, w)$
2: $min\_lat, min\_lon, max\_lat, max\_lon = None$
3: **for all** $node \in sp$ **do**
4: $min\_lat = min(min\_lat, node.min\_lat)$
5: $max\_lat = max(max\_lat, node.max\_lat)$
6: $min\_lon = min(min\_lon, node.min\_lon)$
7: $max\_lon = max(min\_lon, node.min\_lon)$
8: **end for**
9: **if** $slk > 0$ **then**
10: $speed = system\_speed/3.6$ #from km/h to m/s
11: $meters = speed * slk$
12: $earth\_radius = 6378.137$ #km
13: $m = (1/((2 * pi/360) * earth\_radius))/1000$ #1 meter in degree
14: $slk\_min\_lat = min\_lat - (meters * m)$
15: $slk\_max\_lat = max\_lat + (meters * m)$
16: $slk\_min\_lon = min\_lon - (meters * m)/\cos(slk\_min\_lat + pi/180)$
17: $slk\_max\_lon = max\_lon + (meters * m)/\cos(slk\_max\_lat + pi/180)$
18: **end if**
19: $slk\_min\_lat = max(min\_lat, G.min\_lat)$
20: $slk\_max\_lat = min(max\_lat, G.max\_lat)$
21: $slk\_min\_lon = max(min\_lon, G.min\_lon)$
22: $slk\_max\_lon = min(max\_lon, G.max\_lon)$
23: $BBOX = (slk\_min\_lat, slk\_min\_lon, slk\_max\_lat, slk\_max\_lon)$
24: **return** *BBOX*

le *BBOX* pour qu'il ne dépasse pas les limites du graphe étudié (L19-23). Le rectangle englobant étendu *BBOX* représente alors la perception d'un agent entre deux nœuds comme illustré dans la figure 4.1.

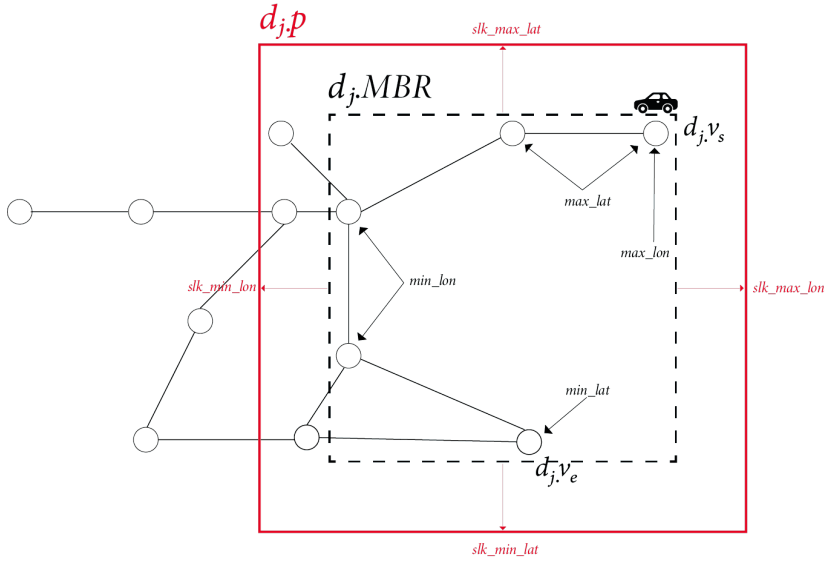

FIGURE 4.1. Exemple de calcul de la perception d'un agent conducteur.

Cette méthode est utilisée pour la perception d'un conducteur, entre chaque arrêt de son ordonnanceur, et pour la perception d'un passager, de son point de départ à son point d'arrivée. Cette perception est approximative, ainsi, un test de faisabilité plus rigoureux, basé sur les contraintes de la section 3.2, est nécessaire avant tout covoiturage.

Un agent conducteur ne met à jour sa perception que dans deux cas : (1) s'il atteint un nœud du graphe de route ; (2) si un contrat de covoiturage impliquant un détour est conclu avec un passager. Un agent passager met à jour sa perception à chaque étape de la simulation. Si un nouveau conducteur apparaît dans sa perception, il cherche à déterminer s'il peut lui permettre de rejoindre sa destination.

4.2. L'identification du covoiturage optimal

Dans nos travaux, nous utilisons les R-Arbres à deux fins. D'une part, les perceptions des agents conducteurs du système sont indexées dans un R-Arbre RT_p par l'agent service de transport *tsa*. Cela permet aux agents passagers d'effectuer des requêtes spatiales à l'agent service de transport, sur des zones spécifiques de l'environnement, et ainsi de restreindre leur espace de recherche. Suite à ces requêtes, l'agent service de transport renvoie les agents conducteurs candidats au covoiturage.

D'autre part, si le covoiturage à évaluer est multi-saut, nous indexons les différents nœuds de transfert possibles dans un autre R-Arbre RT_τ afin de pouvoir transformer le problème en la recherche du plus proche voisin d'un groupe (*GNN/ANN : Group/Aggregate nearest neighbour* en anglais). Ce problème est semblable au problème de l'identification du nœud de transfert optimal auquel nous sommes confrontés. L'objectif est d'identifier le(s) nœud(s) d'un graphe le(s) plus proche(s) d'un groupe d'autres nœuds de ce même graphe. Un exemple simple est le problème du restaurant : quel est le restaurant le plus proche de trois individus éparpillés dans une ville ? Les nœuds de transferts sont alors les restaurants et les conducteurs et le passager sont les individus. La notion du plus proche peut alors être complexe : minimisant la somme des distances (utilitaire), la moyenne des distances (égalitaire)...

Ces opérations sont décrites dans l'algorithme 2, que nous nommons PMO pour *Preferential Matching Optimization*, et l'algorithme 3, que nous nommons CIER pour *Constrained Incremental Euclidean Restriction*.

4.2.1. *L'algorithme d'optimisation préférentielle du covoiturage (PMO)*

Pour établir son espace de recherche, un agent passager r_i effectue trois requêtes spatiales à l'agent *tsa* : une requête Q_2 de chevauchement d'une zone dans l'espace avec sa perception et deux requêtes de chevauchement d'un point dans l'espace : Q_1 sur son origine et Q_3 sur sa destination (BFMO, L1). À la suite de ces requêtes, l'agent passager r_i obtient deux ensembles de candidats conducteurs. Les candidats dont l'itinéraire passe par, ou près, de son nœud de départ et partageant sa perception $candidates\_s = Q_1 \cap Q_2$ et les candidats dont l'itinéraire passe par, ou près, de son nœud d'arrivée et partageant sa perception $candidates\_e = Q_2 \cap Q_3$ (PMO, L2-3).

Algorithm 2 PMO - Preferential Matching Optimization

Input : G = road infrastructure graph, r_i = a rider agent
Output : *best_match* = the best match
1: execute query Q_1, Q_2, Q_3 to service agent
2: *candidates_s* = $Q_1 \cap Q_2$ #candidates for the starting node
3: *candidates_e* = $Q_2 \cap Q_3$ #candidates for the ending node
4: *best_add_dist*, *best_wt*, *best_arrival_time*, *best_nb_tsf* := ∞,
5: *best_matches* := *list*() #$r_i.M$
6: **for all** $d\_s \in candidates\_s$ **do**
7: **for all** $d\_e \in candidates\_e$ **do**
8: **if** $d\_s.id = d\_e.id$ and match is feasible **then**
9: #single hop
10: compute $m.add\_dist = d\_s.ridesharing\_dist - d\_s.initial\_dist$
11: compute $m.total\_wt$, $m.arr\_time$, $m.nb\_tsf$
12: $m.d = d\_s$
13: **else**
14: #multi hop
15: $m = CIER()$
16: **end if**
17: **if** m minimizes all current best values of the objectives (pareto optimal) **then**
18: overwrite *best_matches* := $set(m)$ #the only element of the set is now m
19: update all objectives : *best_add_dist*, *best_wt*, *best_arrival_time*, *best_nb_tsf*
20: **else if** m minimizes at least one objective (pareto dominant) **then**
21: *add* m to *best_matches*
22: update concerned objectives
23: **end if**
24: **end for**
25: **end for**
26: *best_match* = best match among *best_matches* according rider preferences
27: **return** *best_match*

Une fois l'espace de recherche obtenu, l'agent passager doit être capable d'identifier rapidement et efficacement le meilleur covoiturage. Il existe deux possibilités : un covoiturage simple-saut (s-hop), c'est-à-dire un seul conducteur acheminant le passager de son origine à sa destination, et un covoiturage multi-saut (m-hop), c'est-à-dire plusieurs conducteurs (deux dans notre étude) acheminant le passager à sa destination via un transfert. L'agent passager parcourt les deux ensembles de candidats et agit différemment selon le type de covoiturage (PMO, L6-7).

Pour qu'un covoiturage simple-saut soit considéré, un agent conducteur doit appartenir à la fois à l'ensemble *candidats_s* et à l'ensemble *candidats_e* car il doit être candidat au départ et à l'arrivée du passager. Si cette condition est satisfaite et que les contraintes spatio-temporelles définies dans la section 3.2 ne sont pas violées, nous calculons les valeurs des objectifs et générons la solution de covoiturage *m* (PMO, L8-12).

En revanche, si les conducteurs sont différents, alors il s'agit d'un covoiturage multi-saut. Le problème du covoiturage multi-saut est beaucoup plus complexe que celui du simple-saut. En effet, la recherche du nœud de transfert optimal génère de nombreux calculs supplémentaires. Une évaluation de l'ensemble des nœuds de transfert possibles est trop coûteuse computationnellement. C'est pourquoi nous proposons une approche différente, présentée dans l'algorithme 3 nommé CIER (PMO, L15).

4.2.2. *L'algorithme de restriction euclidienne incrémentale sous contraintes (CIER)*

Basé sur l'approche R-Arbres, l'algorithme CIER que nous proposons est une extension de l'algorithme de référence dans la littérature IER [28] pour résoudre le problème GNN/ANN.

Des travaux antérieurs ont mis en évidence le parallèle entre les réseaux routiers sous forme de graphe [7, 32], le problème du covoiturage dynamique multi-saut [31] et le problème du plus proche voisin d'un groupe ou d'un agrégat (d'abord appelé GNN puis ANN) [26]. Si l'on considère T comme l'ensemble des nœuds de transfert disponibles et N comme l'ensemble des nœuds de départ et d'arrivée des utilisateurs, le problème ANN (*aggregate nearest neighbor*) peut être formulé comme suit : une requête ANN permet de retrouver le ou les nœuds de T minimisant la somme des distances par rapport à tous les nœuds de N. En effet, lorsque l'on recherche le nœud de transfert optimal, on cherche le nœud qui minimise la somme des distances de détour des conducteurs associés au covoiturage.

En considérant au maximum un transfert, un transfert est possible s'il y a au moins un nœud de transfert présent dans la perception de l'agent passager r_i et dans la perception d'un conducteur dans *candidates_s* et dans *candidates_e*. Cette zone de rencontre des perceptions des trois agents est nommée la zone de transfert τ (voir figure 4.2) et est calculée comme suit : $\tau = r_i.p \cap d\_s.p \cap d\_e.p$ avec $\tau \subseteq V$, $d\_s \in candidates\_s$ et $d\_e \in candidates\_e$ (CIER, L1). Les nœuds inclus dans τ sont ensuite indexés dans un nouveau R-Arbre RT_τ afin de permettre à nouveau des opérations rapides sur ces données (CIER, L2).

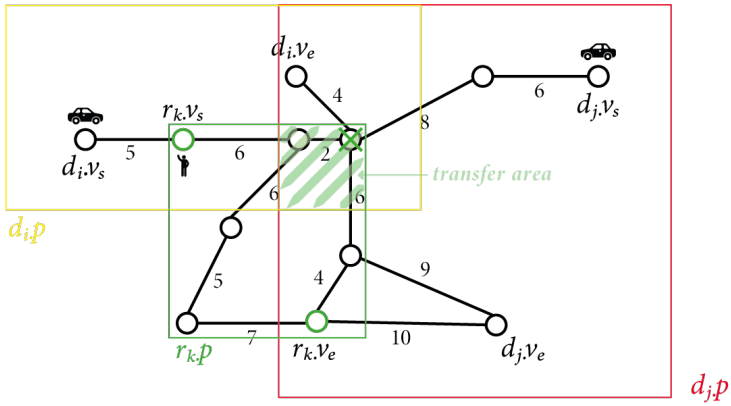

FIGURE 4.2. Identification des candidats au covoiturage à l'aide des différentes perceptions.

Concernant l'ensemble N, six nœuds sont candidats : l'origine et la destination de chacun des participants au covoiturage, r_i, d_s et d_e. Nous avons toutefois fait le choix de ne considérer que quatre de ces nœuds. En effet, deux distances de trajet ne

sont pas impactées par la position du nœud de transfert : celle entre l'origine du premier conducteur d_s et l'origine du passager r_i, et celle entre la destination du passager r_i et la destination du second conducteur d_e. Le nœud d'origine du conducteur candidat au départ et le nœud de destination du conducteur candidat à l'arrivée n'ont donc pas à être considérés dans le problème. Pire encore, ils apporteraient une sorte de « bruit » lors de la minimisation des distances et fausseraient le résultat. En revanche, pour les autres nœuds, la distance de chaque segment dépend de la position du nœud de transfert. Ainsi, $N = \{r.v_s, r.v_e, d\_s.dest, d\_e.origin\}$, avec *origin* et *dest* correspondant respectivement au nœud d'origine et de destination des conducteurs pour la session de covoiturage (CIER, L3). En effet, ces nœuds ne sont pas nécessairement les nœuds de départ v_s et d'arrivée v_e des conducteurs, ils peuvent être des nœuds intermédiaires de leur ordonnanceur de voyage (liste *stops*).

Une fois cette notion de dépendance à la position du nœud de transfert mise en évidence, nous pouvons définir trois types de distances constituant la distance additionnelle de détour d'un covoiturage multi-saut :

- les distances variables de détour ***gnn_dist*** : distances variant en fonction de la sélection du nœud de transfert ;
- les distances fixes initiales ***init_dist*** : distances précalculées initialement prévues par les conducteurs pour rejoindre leur destination sans détours ;
- les distances fixes de détour ***non_gnn_dist*** : distances précalculées liées au détour des conducteurs, mais indépendantes du nœud de transfert.

La distance additionnelle correspond à l'écart entre la distance initiale de trajet des conducteurs, c'est-à-dire la distance sans détours, et la nouvelle distance de trajet. La distance additionnelle de détour à minimiser est donc redéfinie comme suit : $add\_dist = gnn\_dist + non\_gnn\_dist - init\_dist$.

Il existe de nombreuses approches pour résoudre le problème ANN de manière optimale. L'une des plus populaires est l'algorithme IER (*Incremental Euclidean Reduction*) [32]. Il s'agit d'une heuristique de type *Best-First* (le meilleur en premier) basée sur une file d'attente prioritaire (heap queue) H et une indexation des nœuds de transfert dans un R-Arbre. L'algorithme parcourt les différents nœuds de l'arbre en priorisant les éléments les plus proches afin que la première feuille identifiée soit la meilleure. Dans ce chapitre, nous proposons d'étendre cet algorithme en conservant la meilleure distance additionnelle *add_dist* trouvée (la plus petite), à la fois pour le covoiturage à saut unique et à saut multiple. Nous nommons donc cette variable *best_add_dist* et elle est utilisée comme entrée de l'algorithme 3 CIER.

Dans l'algorithme CIER que nous proposons (algorithme 3), nous utilisons principalement les distances euclidiennes pour gérer l'ordre des éléments de la file d'attente prioritaire (CIER, L8-9 & L27-28). En effet, le calcul d'une distance euclidienne entre deux nœuds se fait en $O(1)$ alors que le calcul d'une distance réelle se fait en résolvant le plus court chemin entre ces deux nœuds et est de l'ordre de $O(E + V\log(V))$. Les distances réelles sont bien sûr utilisées plus tard pour vérifier la faisabilité exacte d'une solution.

Algorithm 3 CIER - Constrained Incremental Euclidean Restriction

Input : G : road infrastructure graph, r_i : the rider, d_s : the origin driver, d_e : the destination driver, *best_add_dist* : optimal additional distance
Output : *best_match* : multi-hop match of d_s and d_e with transfer node optimality

1: $\tau = r_i.p \cap d\_s.p \cap d\_e.p$ #transfer area
2: RT_τ := spatial indexing of τ in a rtree
3: $N := \{r.v_s, r.v_e, d\_s.dest, d\_e.origin\}$
4: H := new priority queue
5: *best_match* := *None*
6: compute *non_gnn_dist*, *init_dist*
7: **for all** object o in $RT\_\tau.Root$ **do**
8: compute $add\_dist^{\mathcal{E}} = dist^{\mathcal{E}}_{\text{agg}}(o, N) + non\_gnn\_dist - init\_dist$ #use of euclidean distances
9: enqueue $(H, o, add\_dist^{\mathcal{E}})$
10: **end for**
11: **while** H is not empty **do**
12: o :=dequeue(H)
13: **if** $(add\_dist^{\mathcal{E}} \geqslant best\_add\_dist)$ **then**
14: return *best_match*
15: **end if**
16: **if** o is a leaf (a transfer node) **then**
17: compute $add\_dist = dist_{\text{agg}}(o, N) + non\_gnn\_dist - init\_dist$ #use of real distances
18: **if** $add\_dist < best\_add\_dist$ **then**
19: **if** match is feasible according schedule inserting constraint **then**
20: compute $best\_match.wt$, $best\_match.arr\_time$ and $best\_match.nb\_tsf$
21: $best\_match.v\_tsf = o$, $best\_match.add\_dist = add\_dist$, $best\_match.d_1 = d\_s$, $best\_match.d_2 = d\_e$
22: $best\_add\_dist = add\_dist$
23: **end if**
24: **end if**
25: **else**
26: **for** *child* in o **do**
27: compute $add\_dist^{\mathcal{E}} = dist^{\mathcal{E}}_{\text{agg}}(child, N) + non\_gnn\_dist - init\_dist$
28: enqueue $(H, child, add\_dist^{\mathcal{E}})$
29: **end for**
30: **end if**
31: **end while**
32: **return** *best_match*

Ainsi, au début de l'algorithme 3, nous évaluons la distance euclidienne agrégée $dist^{\mathcal{E}}_{\text{agg}}$ des différents enfants de la racine du R-Arbre RT_τ. Cette distance est calculée à l'aide du MINDIST [1]. Le principe du MINDIST est de calculer la distance euclidienne minimale entre un nœud de $N = \{r.v_s, r.v_e, d\_s.dest, d\_e.origin\}$ et un rectangle englobant du R-Arbre en utilisant les coordonnées longitudinales et latitudinales.

On agrège donc ces distances minimales pour chaque enfant o de la racine du R-Arbre RT_τ (CIER, L6-8). Le total de ces distances constitue la *gnn_dist* du calcul final de la distance additionnelle. On additionne donc les *non_gnn_dist* et on soustrait les *init_dist* précalculées pour obtenir la distance euclidienne additionnelle $add\_dist^{\mathcal{E}}$ (CIER, L8). Chaque enfant est ensuite enfilé dans H et ordonné de façon croissante en fonction de sa distance additionnelle euclidienne (CIER, L9). De cette façon, le rectangle englobant le moins éloigné des nœuds de N sera étudié en premier.

Une fois cette phase d'étude de la racine de l'arbre effectuée, nous rentrons dans le cœur de l'algorithme. Tant que la file H n'est pas vide, l'objet (le rectangle englobant du R-Arbre ou le nœud de transfert) ayant la valeur la plus petite est extrait de la file.

Un premier test est alors effectué quant à la distance euclidienne additionnelle de l'objet extrait de la file. Si cette distance est supérieure à la meilleure distance

additionnelle réelle enregistrée jusqu'ici pour le pas de simulation et l'agent passager, à savoir *best_add_dist*, la boucle est interrompue. En effet, cela veut dire qu'à partir d'ici, on ne trouvera plus de solution améliorante.

S'il est encore possible d'améliorer la solution courante, deux cas se présentent :

- Si l'objet extrait de la file est une feuille du R-Arbre RT_τ, et donc un nœud de transfert, alors nous vérifions si la solution est meilleure en calculant et en considérant la distance additionnelle réelle *add_dist*. Si elle est meilleure et ne viole pas les contraintes de la section 3.2, nous mettons à jour le *best_match* (CIER, L16-23).
- Si l'objet extrait de la file est un nœud du R-Arbre RT_τ, et donc un rectangle englobant, nous effectuons le MINDIST sur ce dernier et l'insérons dans H (CIER, L26-29).

Par conséquent, pour être considérée comme la meilleure association, la distance additionnelle de détour d'un candidat au covoiturage *add_dist* (L17) doit être inférieure à la meilleure distance additionnelle enregistrée *best_add_dist*. L'avantage d'une telle approche est qu'elle permet de réduire considérablement l'espace de recherche en limitant les calculs et l'évaluation des candidats non optimaux. Cependant, il existe un risque de tomber sur un optimum local. En optimisant uniquement la distance de détour supplémentaire, on risque de négliger les objectifs liés aux préférences des passagers. Cette méthode tend à favoriser les agents conducteurs du système en minimisant leurs détours. Elle tend aussi à favoriser le bien commun, en effet, en minimisant les détours, on préserve le patrimoine de détour des conducteurs du système et donc l'offre de transport.

4.2.3. *Une solution de covoiturage personnalisée*

Précédemment, nous avons souligné que l'algorithme CIER favorise la performance de recherche des conducteurs candidats au covoiturage et la minimisation de la distance de détour des agents conducteurs par rapport aux préférences des passagers. Pour répondre à ce constat et diversifier l'offre de covoiturage, la solution que nous proposons est basée sur le mécanisme de dominance de Pareto [29]. L'idée générale est de maintenir une liste des meilleures solutions de covoiturage *best_matches*. Une meilleure solution de covoiturage est une solution minimisant au moins un des objectifs présentés dans la section 3.2 (temps d'attente, heure d'arrivée, distance additionnelle et nombre de transferts). Elle est, dans ce cas, Pareto dominante et est ajoutée à la liste *best_matches*. De cette façon, nous préservons des solutions optimales pour chaque objectif et le passager peut, selon ses préférences, choisir celle qui lui convient le mieux. Si une solution de covoiturage optimise tous les objectifs, alors elle est Pareto optimale, la liste est alors vidée et cette solution lui est ajoutée (PMO, L17-22). Le nombre de solutions optimales est ainsi compris entre 0, quand il n'y a pas de solution, et 4 s'il existe une solution Pareto dominante par objectif.

En plus de cette liste des meilleures solutions, nous maintenons deux autres listes : la liste des pires valeurs *worst_goals* et des meilleures valeurs *best_goals* atteintes

pour chaque objectif. Ces listes servent d'intervalles de référence à un agent passager pour évaluer la performance d'une solution. En effet, en effectuant une différence proportionnelle pondérée par son vecteur de préférence, l'agent passager obtient un pourcentage de performance pour chaque objectif ainsi qu'en moyenne. Ce pourcentage est calculé à l'aide de la formule 4.1 :

$$\frac{\sum_i^k \left(1 - \frac{goals(candidate.goals_i - worst\_goals_i)}{worst\_goals_i - best\_goals_i} * 100\right) * pref_i}{k + sum(pref)} \tag{4.1}$$

Avec k le nombre d'objectifs à considérer. De cette façon, un objectif plus important pour le passager a plus d'impact sur l'évaluation finale de la solution. Le passager sélectionne enfin sa solution préférée (PMO, L28).

5. Expérimentations et résultats

Dans cette section, nous proposons de simuler différents comportements de passagers et de conducteurs à l'aide du modèle agent présenté dans cet article. Les résultats présentés montrent l'impact de ces comportements sur des instances de problème de covoiturage.

5.1. Création des profils utilisateurs

Nous appelons *« un profil utilisateur »* un ensemble de préférences visant à simuler un comportement réel. Pour les conducteurs, nous faisons essentiellement varier la préférence de détour det_{max} entre 0 (c'est-à-dire pas de détour) et 1, correspondant à une capacité de détour de la taille du trajet initial du conducteur. Nous pouvons ainsi générer tout un spectre de comportement de conducteurs, allant du plus égoïste au plus altruiste. Concernant les passagers, nous définissons 5 profils d'utilisateurs présentés dans la Table 5.1 en fonction des 4 objectifs présentés en section 3.2.

Table 5.1. Tableau des différents profils expérimentés pour les agents passagers

Profil	***total_wt***	***add_dist***	***arr_time***	***tsf_nb***
Équilibré	3	3	3	1
Pluie	7	1	1	1
Écologique	1	7	1	1
Pressé	1	1	7	1
Confort	1	1	1	7

Chaque valeur de ce tableau représente le poids associé à chaque objectif : plus le poids est élevé, plus le profil a de l'intérêt pour l'objectif concerné. Concrètement, on peut imaginer un curseur ajustant les poids de ces objectifs dans une application smartphone de covoiturage dynamique. Le profil *« équilibré »* représente un passager

moyen, sans réelles préférences, et sert de profil de référence. Le profil *« pluie »* représente un agent passager sous la pluie qui souhaite être pris en charge rapidement. Il donne alors la priorité à la minimisation du temps d'attente total du trajet. Le profil *« écologiste »* représente un passager préoccupé par l'impact environnemental lié à la conduite d'une voiture. Ce profil privilégie le partage de trajet qui implique peu de détours et cherche donc à minimiser la distance additionnelle de détour liée au covoiturage. Le profil *« pressé »* représente un passager qui souhaite atteindre sa destination le plus rapidement possible. Ce profil donne la priorité à la minimisation de l'heure d'arrivée à sa destination. Enfin, le profil *« confort »* souhaite éviter le désagrément de devoir changer de véhicule au cours d'un trajet. Ce profil donne la priorité aux covoiturages simple-sauts et minimise donc le nombre de transferts de véhicules.

5.2. Protocole expérimental

Nous avons privilégié l'étude de petites instances à de nombreuses reprises afin d'obtenir des résultats reproductibles et fiables. De cette manière, les biais liés aux paramètres du système sont contrôlés. Nous effectuons une simulation avec une génération continue de conducteurs nous permettant de reproduire une situation réelle où il y a un flux entrant et sortant de conducteurs dans le système. En variant la densité de ce flux, nous pouvons simuler tout un panel de trafics routiers et évaluer les comportements en situation de rareté ou d'abondance de l'offre de covoiturage. La résolution de l'instance se termine lorsque tous les agents passagers ont quitté le système : en atteignant leur destination ou en s'impatientant.

Nos instances sont composées d'un graphe de 150 nœuds provenant d'un réseau routier réel de la ville de San Francisco, extrait d'Open Street Map et modélisé à l'aide de la bibliothèque OSMNX [6]. Ce graphe est illustré dans la figure 5.1.

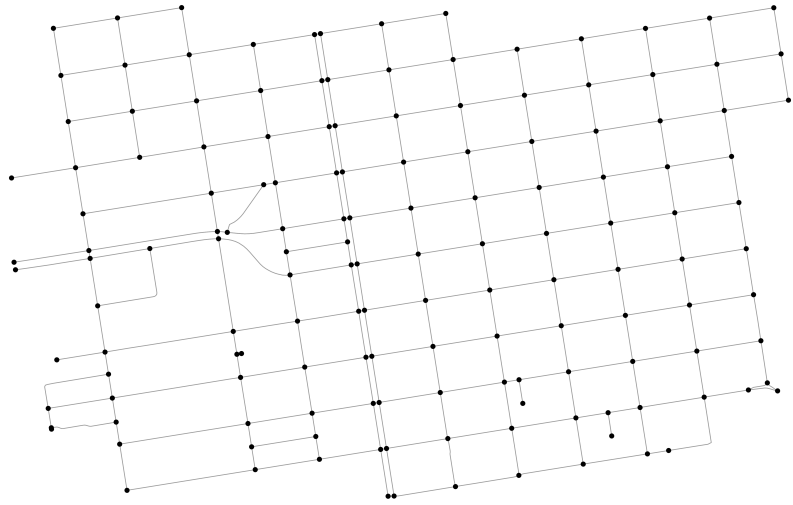

Figure 5.1. Le graphe étudié : une section du côté nord de la ville de San Francisco

Sur ce graphe en grille, 30 conducteurs et 20 passagers interagissent. Leurs nœuds de départ et de destination sont générés de manière aléatoire et uniforme. Les itinéraires initiaux des conducteurs sont dérivés du plus court chemin entre leurs nœuds de départ et d'arrivée. Nous supposons que les passagers ont un temps d'attente maximal de 3 minutes et un facteur de détour fixé à 0,2 pour créer une situation complexe.

5.3. Résultats

Dans cette sous-section, nous détaillons les résultats obtenus avec notre modèle. L'objectif est de mettre en évidence l'impact des différentes stratégies sur des indicateurs tels que : le temps d'attente moyen, le taux de service (succès du covoiturage), la distance additionnelle moyenne et le nombre moyen de véhicules nécessaires au covoiturage. Chaque figure comprend plusieurs courbes correspondant aux profils des passagers mentionnés précédemment. Chaque point d'une courbe est le résultat de 50 expériences moyennées. Ces expériences correspondent aux 50 mêmes expériences pour chaque profil de passager. Enfin, l'axe des abscisses est indexé par le facteur de détour des conducteurs.

5.3.1. *L'impact du profil « écologiste »*

La figure 5.2 montre comment le profil « écologiste » (courbe verte/triangles) minimise la distance additionnelle moyenne de détour des conducteurs par rapport aux autres profils. Ce phénomène est amplifié par l'augmentation du facteur de détour des conducteurs. On observe ainsi des détours 2 à 3 fois moins importants avec ce profil qu'avec les profils concurrents. En conséquence, les autres objectifs sont très largement négligés par ce profil (figures 5.3, 5.4, 5.5).

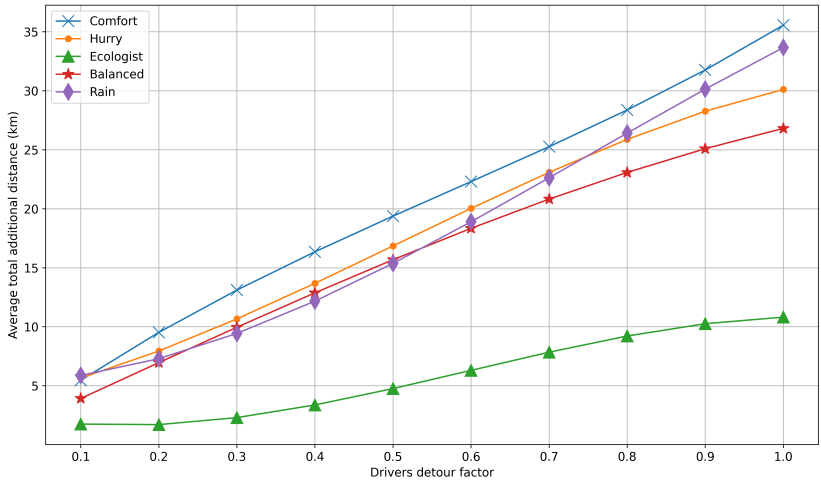

Figure 5.2. Distance additionnelle moyenne de détour des conducteurs

5.3.2. *L'impact du profil « confort »*

Le profil « confort » (courbe bleue/croix) dans la figure 5.3, vise à minimiser le nombre de transferts (changements de véhicules) pour atteindre la destination cible. On constate que ce profil est capable de minimiser sa cible de plus en plus (environ de 1,4 à 1,1 véhicule impliqué en moyenne) avec l'augmentation du facteur de détour des conducteurs. Il est également bien meilleur que les autres profils en montrant entre 20 % et 70 % de performance supplémentaire sur cet objectif. En revanche, il est le plus faible pour la distance de détour supplémentaire (figure 5.2) et est similaire à la majorité des autres profils pour les autres objectifs (figures 5.4 et 5.5). Nous pouvons conclure ici que la réduction du nombre de transferts entre véhicules et la favorisation du covoiturage simple-saut augmente la distance de détour.

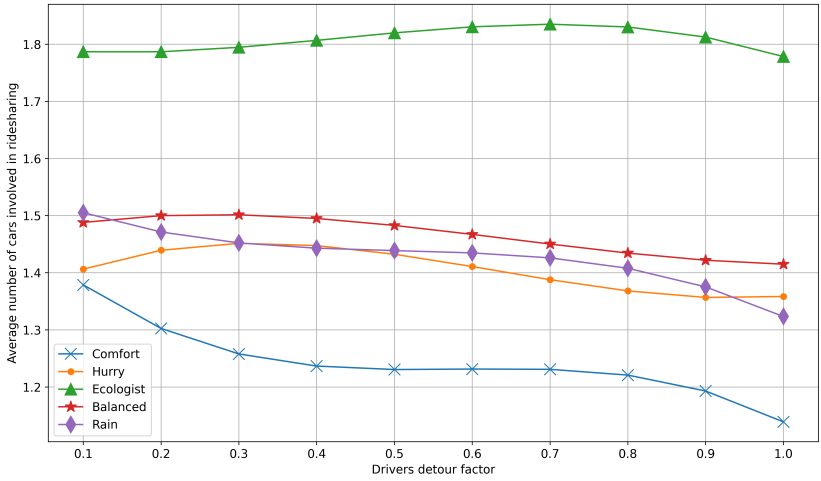

FIGURE 5.3. Nombre moyen de véhicules impliqués dans le covoiturage

5.3.3. *L'impact des profils « pluie », « pressé » et « équilibré »*

Nous avons choisi de traiter ces trois profils conjointement, car, dans la majorité des cas, la tendance des courbes et leurs résultats sont similaires. Cela s'explique par la difficulté de dissocier le temps de parcours, le temps d'attente et la distance de parcours. Ces derniers sont en effet étroitement liés et interdépendants. Néanmoins, le profil « pluie » (courbe mauve/losanges) réussit à réduire son temps d'attente (figure 5.4) quand les conducteurs sont peu disposés à effectuer des détours (facteur inférieur à 0,5). Il se confond ensuite avec les autres profils.

Le profil « Pressé », cherchant à arriver le plus rapidement possible, vise indirectement à réduire son temps d'attente. C'est pourquoi nous le voyons se positionner très légèrement comme le leader de la minimisation du temps d'attente sur la figure 5.4

pour un facteur de détour du conducteur de 0,5 à 0,9. Notre profil « équilibré » remplit parfaitement son objectif d'être le juste milieu entre les différents profils et leurs objectifs, comme on peut le voir sur la totalité des figures.

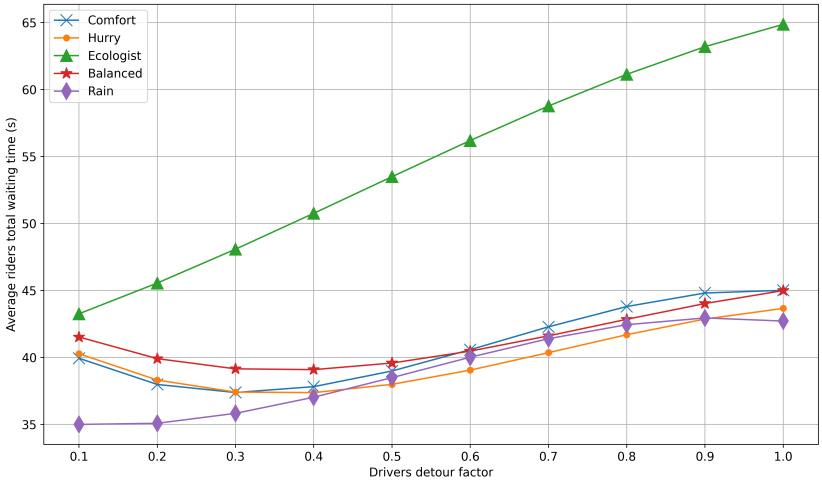

FIGURE 5.4. Temps d'attente moyen d'un passager

5.3.4. *La performance globale des profils : le taux de service*

Pour conclure sur l'analyse de ces résultats, nous pouvons considérer la performance des profils du point de vue du taux de service moyen (figure 5.5). Le taux de service représente la part des passagers qui ont réussi à se rendre à leur destination en covoiturant.

On constate tout de suite que plus le facteur de détour augmente, plus il existe de solutions potentielles et donc plus le taux de service augmente. Ce phénomène est normal et démontre le bon fonctionnement de notre modèle. Nous remarquons que le profil « écologiste » (triangles verts) présente un taux de service relativement faible par rapport au reste des profils avec une différence moyenne de 5 à 15 %. Aussi, il représente le pire taux de service atteint avec environ 72 % pour un facteur de détour des conducteurs de 1. Si on émet l'hypothèse que les utilisateurs qui souhaitaient être passagers ont finalement pris leur véhicule pour rejoindre leur destination, il se peut que ce profil ne permette pas de réduire, autant qu'espéré, les émissions globales du système, curieux paradoxe !

6. CONCLUSION ET PERSPECTIVES

Dans cet article, nous proposons un système multi-agent pour optimiser le covoiturage dynamique en fonction des préférences des conducteurs et des passagers. Nous modélisons les différents acteurs d'un système de covoiturage et leur spectre de comportements possibles. Nous apportons une approche originale, basée sur des techniques

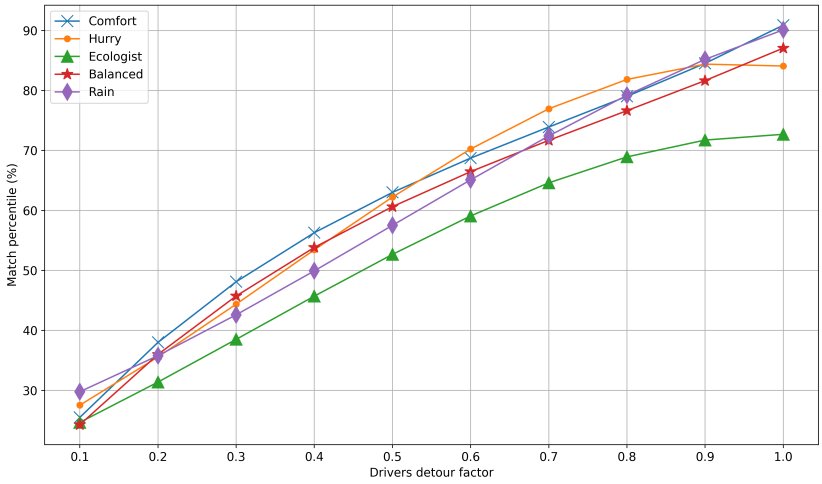

FIGURE 5.5. Taux de service moyen du système

d'indexation spatiale, pour modéliser la perception de nos agents et pour réduire le coût computationnel associé à la recherche du covoiturage optimal. Les expériences présentées ont montré que l'approche proposée permet une optimisation individuelle et efficace.

Cette approche révèle certaines limitations pour les profils favorisant le temps d'attente et le temps de trajet. En effet, l'optimisation par agrégation des pondérations semble insuffisante lorsque les objectifs sont interdépendants. Pour répondre à ce constat, d'autres techniques peuvent être envisagées, comme l'agrégation par l'intégrale de Choquet, qui s'est avérée efficace pour l'optimisation multicritère [5]. Une autre perspective consiste à rendre le comportement des agents dynamique. En effet, le comportement des agents est pour l'instant défini à leur apparition et ne change pas au cours du temps. En fonction de l'état de leur environnement (trafic, météo, pénurie d'offres ou de demandes, etc.), leur comportement pourrait évoluer et les mener à changer leurs objectifs principaux [3]. En étendant notre approche à des agents au comportement dynamique, nous nous rapprocherions encore davantage du monde réel et de ses caractéristiques.

Des critères et mécanismes tels que le nombre d'arrêts maximum souhaité par un conducteur ou encore la possibilité de réserver un seul covoiturage pour plusieurs personnes sont essentiels dans le cadre d'une mise en application de notre solution. Ils seront inclus et étudiés dans nos prochains articles.

Enfin, il serait fructueux de considérer également la complémentarité avec d'autres modalités de transport, en particulier les transports publics tels que le bus et le métro [12, 20]. Nous passerions alors de la mono-modalité à la comodalité, c'est-à-dire la multimodalité associée au covoiturage.

Bibliographie

[1] M. Adler & B. Heeringa, « Search Space Reductions for Nearest-Neighbor Queries », in *Theory and Applications of Models of Computation* (M. Agrawal, D. Du, Z. Duan & A. Li, éds.), vol. 4978, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, p. 554-567.

[2] N. Agatz, A. L. Erera, M. W. Savelsbergh & X. Wang, « Dynamic Ride-Sharing: A Simulation Study in Metro Atlanta », *Procedia – Social and Behavioral Sciences* **17** (2011), p. 532-550.

[3] F. Balbo & S. Pinson, « Dynamic Modeling of a Disturbance in a Multi-Agent System for Traffic Regulation », *Decision Support Systems* **41** (2005), nº 1, p. 131-146.

[4] H. A. N. C. Bandara & D. Dias, « A Multi-Agent system for dynamic ride sharing », in *2009 International Conference on Industrial and Information Systems (ICIIS)*, IEEE, 2009, p. 199-203.

[5] S. Ben Cheikh-Graiet, M. Dotoli & S. Hammadi, « A Tabu Search Based Metaheuristic for Dynamic Carpooling Optimization », *Computers & Industrial Engineering* **140** (2020), article no. 106217.

[6] G. Boeing, « OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks », *Computers, Environment and Urban Systems* **65** (2017), p. 126-139.

[7] Z. Chen, B. Yao, Z.-J. Wang, X. Gao, S. Shang, S. Ma & M. Guo, « Flexible Aggregate Nearest Neighbor Queries and its Keyword-Aware Variant on Road Networks », *IEEE Transactions on Knowledge and Data Engineering* **33** (2021), nº 12, p. 3701-3715.

[8] J.-F. Cordeau & G. Laporte, « The Dial-a-Ride Problem: Models and Algorithms », *Ann Oper Res* **153** (2007), p. 29-46.

[9] C. Cortés, M. Matamala & C. Contardo, « The Pickup and Delivery Problem with Transfers », *Eur. J. Oper. Res.* **200** (2010), nº 3, p. 711-724.

[10] A. Daoud, F. Balbo, P. Gianessi & G. Picard, « Un Modèle Agent Générique Pour La Comparaison d'approches d'allocation de Ressources Dans Le Domaine Du Transport à La Demande », in *JFSMA29*, Cépaduès, 2021, p. 127-136.

[11] A. Di Febbraro, E. Gattorna & N. Sacco, « Optimization of Dynamic Ridesharing Systems », *Transportation Research Record: Journal of the Transportation Research Board* **2359** (2013), nº 1, p. 44-50.

[12] A. O. Diallo, G. Lozenguez, A. Doniec & R. Mandiau, « Usage Des Parkings Relais Dans Les Comportements de Déplacements Intermodaux », in *JFSMA29, June 28-30, 2021*, Cépaduès, 2021, p. 83-92.

[13] J. Ferber, « Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence », 1999.

[14] C. Fevre, H. Zgaya-Biau, P. Mathieu & S. Hammadi, « Multi-Agent Systems and R-Trees for Dynamic and Optimised Ridesharing », in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, p. 1352-1358.

[15] A. Guttman, « R-trees: a dynamic index structure for spatial searching », vol. 14, SIGMOD Rec., nº 2, Association for Computing Machinery, New York, NY, USA, 1984, p. 47-57.

[16] W. Herbawi & M. Weber, « Evolutionary Multiobjective Route Planning in Dynamic Multi-hop Ridesharing », in *Evolutionary Computation in Combinatorial Optimization*, Springer, Berlin, Heidelberg, 2011, p. 84-95.

[17] ———, « Modeling the Multihop Ridematching Problem with Time Windows and Solving It Using Genetic Algorithms », *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI* **1** (2012), p. 89-96.

[18] K. Jeribi, H. Mejri, H. Zgaya & S. Hammadi, « Distributed Graphs for Solving Co-Modal Transport Problems », in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, p. 1150-1155.

[19] E. Kamar & E. Horvitz, « Collaboration and Shared Plans in the Open World: Studies of Ridesharing », in *IJCAI International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, p. 187-194.

[20] J. Lin, S. Sasidharan, S. Ma & O. Wolfson, « A Model of Multimodal Ridesharing and Its Analysis », in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, IEEE, 2016, p. 164-173.

[21] S. Ma & O. Wolfson, « Analysis and Evaluation of the Slugging Form of Ridesharing », in *SIGSPATIAL'13*, ACM Press, Orlando, Florida, 2013, p. 64-73.

[22] S. M. Ma, Y. Zheng & O. Wolfson, « T-Share: A Large-Scale Dynamic Taxi Ridesharing Service », in *ICDE29'*, IEEE, 2013, p. 410-421.

[23] Y. Manolopoulos, Y. Theodoridis & V. J. Tsotras, « Spatial Indexing Techniques », in *Encyclopedia of Database Systems*, Springer New York, New York, NY, 2014, p. 1-7.

[24] N. Masoud & R. Jayakrishnan, « A Real-Time Algorithm to Solve the Peer-to-Peer Ride-Matching Problem in a Flexible Ridesharing System », *Transportation Research Part B: Methodological* **106** (2017), p. 218-236.

[25] P. Mathieu & A. Nongaillard, « Effective Evaluation of Autonomous Taxi Fleets », in *Proceedings of the 10th International Conference on Agents and Artificial Intelligence – Volume 1: ICAART,*, SciTePress, 2018, p. 297-304.

[26] D. Papadias, Q. Shen, Y. Tao & K. Mouratidis, « Group nearest neighbor queries », in *Proceedings. 20th International Conference on Data Engineering*, 2004, p. 301-312.

[27] D. Papadias, Y. Tao, K. Mouratidis & C. K. Hui, « Aggregate Nearest Neighbor Queries in Spatial Databases », *ACM Transactions on Database Systems* **30** (2005), n° 2, p. 529-576.

[28] D. Papadias, J. Zhang, N. Mamoulis & Y. Tao, « Query Processing in Spatial Network Databases », in *Proceedings 2003 VLDB Conference*, Elsevier, 2003, p. 802-813.

[29] V. Pareto, *Cours d'économie politique*, Librairie Droz, 1964.

[30] A. Tafreshian & N. Masoud, « Trip-Based Graph Partitioning in Dynamic Ridesharing », *Transportation Research Part C: Emerging Technologies* **114** (2020), p. 532-553.

[31] Y. Xu, L. Kulik, R. Borovica-Gajic, A. Aldwyish & J. Qi, « Highly Efficient and Scalable Multi-hop Ride-sharing », in *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, ACM, Seattle WA USA, 2020, p. 215-226.

[32] M. Yiu, N. Mamoulis & D. Papadias, « Aggregate Nearest Neighbor Queries in Road Networks », *IEEE Transactions on Knowledge and Data Engineering* **17** (2005), p. 820-833.

Abstract. — In this paper, we propose a multi-agent approach to solve the dynamic multi-hop ridesharing problem. In our system, passengers and drivers are represented as autonomous and rational agents in perpetual interaction to satisfy their own objectives such as their waiting time or their travel time. In the proposed solution, driver and passenger agents have a dynamically modeled perception using R-Trees. We model their detour and route preferences and show the impact of these on the resolution of a dynamic ridesharing instance. The presented results show that our system dynamically handles complex passenger requests while minimizing the impact of ridesharing for drivers across a wide spectrum of preferences and behaviors.

Keywords. — Ridesharing, Simulation, Optimization, Agents.
