Coordination in Connected Autonomous Vehicle fleets : A Multiagent Resource Allocation Approach to Online On-Demand Transport

Alaa Daoud

N°d'ordre NNT : 2022LYSEM002

THESE de DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'École des Mines de Saint-Étienne

École Doctorale N°488
Science, Ingénierie et Santé

Spécialité de doctorat: Informatique
Discipline: Intelligence Artificielle et Systèmes Multiagents

Soutenue le 17/01/2022, par:

Alaa DAOUD

Coordination de flottes de véhicules autonomes connectés :
Approche décentralisée d'allocation de ressources pour le transport à la demande

Devant le jury composé de:

Feillet, Dominique	Professeur	LIMOS, École des Mines de Saint-Étienne	Président
Klügl, Franziska	Professeur	Orebro University	Rapporteur
Zargayouna, Mahdi	Ph.D. HDR	GRETTIA, Université Gustave Eiffel	Rapporteur
Gleizes, Marie-Pierre	Professeur	IRIT, Université Toulouse	Examinatrice
Balbo, Flavien	Professeur	LIMOS, École des Mines de Saint-Étienne	Directeur de thèse
Picard, Gauthier	Senior Research Fellow, PhD, HDR	ONERA/DTIS, Université de Toulouse	Co-directeur de thèse
Gianessi, Paolo	Maître Assistant	LIMOS, École des Mines de Saint-Étienne	Encadrant

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	M. Batton-Hubert
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	SCIENCES DES IMAGES ET DES FORMES	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	N. Absi, Maitre de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	V.Laforest, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	MR	Génie industriel	CMP
AUGUSTO	Vincent	MR	Génie industriel	CIS
AVRIL	Stéphane	PR	Mécanique et ingénierie	CIS
BADEL	Pierre	PR	Mécanique et ingénierie	CIS
BALBO	Flavien	PR	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR	Mathématiques appliquées	FAYOL
BEIGBEDER	Michel	MA	Informatique	FAYOL
BILAL	Blayac	DR	Sciences et génie de l'environnement	SPIN
BLAYAC	Sylvain	PR	Microélectronique	CMP
BOISSIER	Olivier	PR	Informatique	FAYOL
BONNEFOY	Olivier	PR	Génie des Procédés	SPIN
BORBELY	Andras	DR	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR	Génie Industriel	FAYOL
BRUCHON	Julien	PR	Mécanique et ingénierie	SMS
CAMEIRAO	Ana	PR	Génie des Procédés	SPIN
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR	Génie Industriel	CMP
DEBAYLE	Johan	MR	Sciences des Images et des Formes	SPIN
DEGEORGE	Jean-Michel	MA	Génie industriel	Fayol
DELAFOSSE	David	PR	Sciences et génie des matériaux	SMS
DELORME	Xavier	PR	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
BERGER-DOUCE	Sandrine	PR	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR	Mécanique et ingénierie	SMS
DUTERTRE	Jean-Max	PR	Microélectronique	CMP
EL MRABET	Nadia	MA	Microélectronique	CMP
FAUCHEU	Jenny	MA	Sciences et génie des matériaux	SMS
FAVERGEON	Loïc	MR	Génie des Procédés	SPIN
FEILLET	Dominique	PR	Génie Industriel	CMP
FOREST	Valérie	PR	Génie des Procédés	CIS
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GAVET	Yann	MA	Sciences des Images et des Formes	SPIN
GERINGER	Jean	MA	Sciences et génie des matériaux	CIS
GONDRAN	Natacha	MA	Sciences et génie de l'environnement	FAYOL
GONZALEZ FELIU	Jesus	MA	Sciences économiques	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GRIMAUD	Frederic	EC	Génie mathématiques et industriel	FAYOL
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR	Génie des Procédés	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR	Génie des Procédés	SPIN
ISMAILOVA	Esma	MC	Microélectronique	CMP
KERMOUCHE	Guillaume	PR	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	DR	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	DR	Mécanique et ingénierie	FAYOL
LIOTIER	Pierre-Jacques	MA	Mécanique et ingénierie	SMS
MEDINI	Khaled	EC	Sciences et génie de l'environnement	FAYOL
MOLIMARD	Jérôme	PR	Mécanique et ingénierie	CIS
MOULIN	Nicolas	MA	Mécanique et ingénierie	SMS
MOUTTE	Jacques	MR	Génie des Procédés	SPIN
NAVARRO	Laurent	MR	Mécanique et ingénierie	CIS
NEUBERT	Gilles	PR	Génie industriel	FAYOL
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
O CONNOR	Rodney Philip	PR	Microélectronique	CMP
PICARD	Gauthier	PR	Informatique	FAYOL
PINOLI	Jean Charles	PR	Sciences des Images et des Formes	SPIN
POURCHEZ	Jérémy	DR	Génie des Procédés	CIS
ROUSSY	Agnès	MA	Microélectronique	CMP
SANAUR	Sébastien	MA	Microélectronique	CMP
SERRIS	Eric	IRD	Génie des Procédés	FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
VALDIVIESO	François	PR	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR	Génie industriel	CIS
YUGMA	Gallian	MR	Génie industriel	CMP

Mise à jour : 04/02/2021

Coordination in Connected Autonomous Vehicle fleets:
A Multiagent Resource Allocation Approach to Online On-Demand Transport

Coordination de flottes de véhicules autonomes connectés :
Approche décentralisée d'allocation de ressources pour le transport à la demande

PhD DISSERTATION OF THE UNIVERSITY OF LYON
operated within
École des Mines de Saint-Étienne

École Doctorale ED SIS n°488
Science, Ingénierie et Santé

Doctoral specialization: Computer Science
Discipline: Artificial Intelligence and Multiagent Systems

Dissertation prepared by:
Alaa DAOUD

Univ Lyon, LIMOS UMR CNRS 6158,
Institut Henri Fayol, Mines Saint-Étienne, F-42023 Saint-Étienne, France.

Composition of the Jury

Feillet, Dominique	Professor	LIMOS, École des Mines de Saint-Étienne	President
Klügl, Franziska	Professor	Orebro University	Reviewer
Zargayouna, Mahdi	Ph.D. HDR	GRETTIA, Université Gustave Eiffel	Reviewer
Gleizes, Marie-Pierre	Professor	IRIT, Université Toulouse	Examiner
Balbo, Flavien	Professor	LIMOS, École des Mines de Saint-Étienne	Supervisor
Picard, Gauthier	Senior Research Fellow, PhD, HDR	ONERA/DTIS, Université de Toulouse	Supervisor
Gianessi, Paolo	Associate Professor	LIMOS, École des Mines de Saint-Étienne	Supervisor


Institut Mines-Télécom

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE L'ÉCONOMIE
DE L'INDUSTRIE ET DU NUMÉRIQUE

cnrs

LIMOS

To Clara ...

"You'll have such proof as exists. You are the only one responsible for your own wants."

— Isaac Asimov, I, Robot

Cette thèse est rédigée en anglais. Cependant, pour faciliter son accessibilité aux francophones, nous fournissons en français une traduction de l'introduction (page xv), d'une synthèse des parties clés de la thèse (Annex C page 151), et une traduction de la conclusion (Annex D page 167).

This dissertation is written in English. However, to facilitate its accessibility to french speaking people, we provide in French a translation to the introduction (page xv), a summary of the key points of the thesis (Appendix C page 151), and a translation to the conclusion (Appendix D page 167).

Acknowledgments

The past forty months have been an extraordinary journey. Well, except that the most important part of my work on this dissertation concurrent with the epidemic of Covid-19 with all its subsequent economic, social and psychological effects. Still, I finally succeeded in fulfilling the requirements to defend this thesis. For those who were and are still on the front lines keeping us going despite the pandemic, no words are really sufficient to thank you.

First of all, I would like to thank the members of the jury, Prof. Marie-Pierre Gleize as examiner and Prof. Dominique Feillet as president for accepting to evaluate my work and for their valuable questions and comments. I would also like to thank Prof. Franziska Klügl and Prof. Mahdi Zargayouna for their valuable reviews, mostly positive, but also for pointing out some weak points.

Of course, I would like to express my thanks to my supervisors for their valuable advice and patience during my doctoral studies.

To Prof. Flavien Balbo, for his invaluable support throughout the project on many aspects, scientific, personal and administrative.

To Prof. Gauthier Picard, whose revisions of my scientific and English writings were, I believe, the most critical and accurate.

To Associate Prof. Paolo Gianessi who made enormous efforts to correct my mathematical formulations and jargon.

It has been a great adventure and a long journey during which I have enjoyed working with you. I have learned many things in addition to those directly related to my field of research. I learned about scientific writing, peer review, as well as socialising in the scientific community.

Talking about socializing always reminds about the consequences of the epidemic. This epidemic that shows how fragile is mankind, taught us how essential is social life, how indispensable is communication and most significantly how important is family.

I'd like to thank my family, my mother who taught me to be persistent, and my father who taught me to be strong and face life's difficulties through hard work.

Samer, my brother that I'm glad to have him behind me in person, for being with me in all important moments of my life.

My brother Diaa, the young successful guy, beloved from everybody because of his funny, strong, and supportive personality.

All members of my grand family, and my family-in-low for their unconditional support despite all the challenges that they face in Syria.

I'd like to thank my cousin Salma, and those who I consider members of my extended family: Dr. Ismaail Alouch, Mona Alshaar, and their son, Fabio for being next to me.

My dear friends and coworkers, Dr. Omar Alqawasmeh, Dr. Amro Najjar, Dr. Yazan Mualla, Mohammad Badr, and Mohammad Kassem.

My colleagues members and ex-members of ISI and GMI departments at Fayol Institute - Mines Saint-Etienne, with whom I shared great working time.

I'd like to express special thanks to Prof. Olivier Boissier, the director of this institute for his beloved personality and his continuous intention to help in all aspects, and also to Marie-Line Barneoud, Jean-François Tchebanoff and Niloufare Sadr for all technical and administrative assistance.

To all my friends, colleagues and teachers from all over the world, thank you for being lovely parts of my life.

Finally, to the most important person in my life, who gave me support, love, and attention in my weakness and strength. To my spouse and partner in every moment Dr. Hiba Alqasir I say: without you, I wouldn't have been here today, and yes we can as we are together, and we'll never stop dreaming as we keep acting towards getting these dreams into reality. I dedicate this success to you and to our love that grows day by day in front of our eyes. You are and have always been my "Hiba",[1] you'll stay as always the Jasmine rose of my life.

[1] "Hiba": in Arabic is a female name that literally means gift

Abstract

The development of autonomous vehicles, capable of peer-to-peer communication, as well as the interest in on-demand solutions (e.g., Uber, Lyft, Heetch), are the primary motivations for this study. More precisely, we are interested here in solving the problem of allocating autonomous vehicles in a decentralized manner. A fleet of autonomous vehicles is deployed to respond to numerous requests from different locations in the city. Typically, this problem is solved by centralizing the requests in a portal where a fleet manager assigns them to vehicles. This implies that the vehicles have continuous access to the portal (via a cellular network, for example). However, accessing such a global switching infrastructure (for data collection and order delivery) is costly and represents a critical bottleneck. The idea is to use low-cost vehicle-to-vehicle (V2V) communication technologies to allow vehicles to coordinate without a global communication infrastructure. We propose to model the different aspects of decision and optimization problems related to this more general challenge. Then, the question arises as to the choice between centralized and decentralized solution methods. Methodologically, we explore the directions and compare the performance of distributed constraint optimization techniques (DCOP), self-organized multiagent techniques, market-based approaches, and centralized operations research solutions.

Keywords

Multiagent, On demand Transport, Resource Allocation, Connected Autonomous Vehicles, Simulation.

Contents

List of Figures

List of Tables

Introduction

Les premiers essais connus de conception de véhicules sans conducteur remontent à 500 ans, à une époque où les véhicules motorisés n'étaient même pas une lueur d'espoir. Aux alentours de 1478, Léonard de Vinci a dessiné les plans du premier véhicule autopropulsé au monde [57]. Son véhicule (charrette) n'était pas destiné à transporter des passagers à bord. Il était capable de se déplacer et de changer de direction sans être poussé. Il était propulsé par des ressorts hélicoïdaux. Il était également doté d'une direction programmable, obtenue en disposant des blocs de bois entre des engrenages à des endroits prédéfinis. Curieusement, elle ne pouvait tourner qu'à droite. À la fin des années 1960, John McCarthy, dans un essai à Stanford, imagine qu'un véhicule automatisé pourrait un jour le conduire à l'aéroport [109]. En 1987, Dickmanns and Zapp ont fait la démonstration d'une camionnette Mercedes autopilotée sur l'autoroute allemande (autobahn) ; elle pouvait rester dans les voies et dépasser [46]. Cependant, ces exemples sont considérés comme automatisés, alors que plus de 30 ans plus tard, l'autonomie n'est pas encore totalement atteinte [134].

Les véhicules modernes (à partir du niveau 2 de la conduite automatisée définie par SAE[2]) ont été équipés de nombreux capteurs internes au cours des dernières décennies : niveau de carburant, température du moteur, niveau de la batterie, rappel de ceinture de sécurité, etc. Ces capteurs sont principalement destinés à surveiller l'état de sécurité du véhicule. Avec l'avènement de la mobilité coopérative, connectée et automatisée (CCAM), des capteurs externes tels que des radars et des caméras ont été introduits. L'objectif de ces capteurs est de détecter la présence et le comportement des autres usagers du transport. En combinant plusieurs capteurs et en appliquant un modèle de reconnaissance d'objet, les véhicules équipés de ces capteurs peuvent collecter des données (ils "voient" et "entendent" pour construire leurs modèles de connaissance du monde environnant). Grâce à la connectivité des véhicules, ces données peuvent être partagées avec d'autres.
Au cours de la dernière décennie, des technologies de communication entre véhicules à courte et longue portée ont été développées et introduites dans le domaine des transports dans le but principal d'améliorer la sécurité et l'efficacité du trafic. Les technologies de communication entre véhicules comprennent des équipements, des applications et des systèmes permettant la communication de véhicule à véhicule (V2X). Ces capacités ouvrent la voie au domaine d'application de la conduite coopérative. En plus d'être des entités autonomes individuelles, des groupes de véhicules (tels que ceux appartenant à la même entreprise) peuvent se comporter comme des flottes et bénéficier de leur intelligence collective, collecter des informations, les partager entre eux, s'adapter aux conditions du trafic et de l'environnement environnant, pour atteindre leurs objectifs communs. Dans cette étude, nous nous intéressons au comportement coopératif en tant que technologie permettant de résoudre le problème du transport à la demande. Ce problème est

[2]SAE signifie Society of Automotive Engineers, une association professionnelle basée aux États-Unis et active dans le monde entier, ainsi qu'un organisme de développement de normes pour les professionnels de l'ingénierie dans divers secteurs. Créée au début des années 1900 sous le nom de Society of Automobile Engineers

également connu sous le nom de *Dial-a-Ride Problem*, dans lequel nous disposons d'une flotte de véhicules distribués en différents endroits de la ville pour répondre aux demandes des passagers qui souhaitent se déplacer entre les lieux de prise en charge et d'arrivée en respectant un ensemble de contraintes sur la demande.

Motivation

Les systèmes de transport évoluent rapidement, en raison de développements technologiques tels que l'intelligence artificielle et les technologies de communication émergentes (par exemple, la 5G cellulaire, les protocoles sans fil LP-WAN et les communications ad hoc entre véhicules via les protocoles ITS-G5). En outre, l'évolution des préférences des entreprises et des clients met en évidence le besoin de systèmes de mobilité personnalisés et intelligents. Ces nouvelles solutions de mobilité peuvent avoir la capacité de modifier fondamentalement le domaine des transports. Par exemple, le transport à la demande (ODT)[3] peut conduire à la transition de la propriété d'un véhicule à un véhicule en tant que service, ce qui réduirait le nombre de véhicules, tout en augmentant l'efficacité du transport et en réduisant ainsi les mouvements inutiles, tout en conservant le même niveau de qualité de service. Ces avantages potentiels auraient des répercussions environnementales, sociales et économiques [54]. Le développement de flottes de véhicules autonomes, capables de communiquer de pair à pair, ainsi que l'intérêt pour les solutions à la demande (par exemple, Uber, Lyft, Heetch) sont les principales motivations de cette recherche. Nous souhaitons étudier le problème de la mise en place d'une flotte de véhicules autonomes capables de répondre dynamiquement, en l'absence de contrôle central, à des demandes de déplacement en ligne dans une ville. La vision de l'allocation de ressources multi-agents (MARA) est pertinente pour un large éventail de domaines d'application, y compris le transport. Le domaine multi-agent est bien adapté à la modélisation et au développement de systèmes décentralisés. Par conséquent, l'allocation de véhicules est un domaine d'application pertinent pour les techniques multi-agents [49, 100, 137, 143]. En revanche, la centralisation du processus d'allocation avec un répartiteur automatique est encore assez courante dans les approches multi-agents [49, 143, 102].

Défi 1. Antagonisme des objectifs dans l'allocation de ressources

Les problèmes d'allocation sont des questions majeures dans la gestion des systèmes ODT. Ils sont étudiés depuis des décennies et diverses solutions ont été proposées. En pratique, la faisabilité et l'efficacité du choix de centraliser ou de décentraliser les méthodes de résolution dépendent de la complexité du problème, de ses contraintes et de la dynamique de l'environnement. Cependant, même les cas les plus simples d'allocation de ressources conduisent à des débats sur l'efficacité versus l'équité [98], l'efficacité versus la vie privée [12] et sur la qualité de l'entreprise versus l'expérience de l'utilisateur [140].

Défi 2. Une solution adaptative dynamique

Les solutions aux problèmes d'allocation des ressources dans les contextes dynamiques des systèmes de télétravail doivent remettre en question les horaires des véhicules en temps réel. Ce défi fait que l'obtention d'une solution globalement optimale est un objectif difficile à atteindre dans la pratique. Cependant, la conception d'approches d'amélioration pour des solutions réalisables est une alternative appropriée pour traiter les problèmes de la dimension dynamique du problème : cela nécessite de prendre en compte l'aspect communication et de fournir des mécanismes de communication et de coordination robustes et efficaces.

[3] Acronyme du terme anglais "On-Demand Transport".

Défi 3. Goulot d'étranglement de la communication

L'un des principaux problèmes liés à l'utilisation des systèmes multi-agents (SMA) et des approches d'allocation de ressources multi-agents (MARA) pour résoudre les problèmes liés aux ODT est le goulot d'étranglement de la communication. Selon Jin and Jie, chaque agent n'a besoin de communiquer qu'avec un ensemble limité de voisins dans sa zone de planification au lieu de communiquer avec tous ses pairs [81]. Le développement de solutions décentralisées basées sur les interactions de véhicule à véhicule (V2V) pourrait être une source essentielle d'économies et de résilience.

Notre thèse est que nous pouvons relever les défis ci-dessus en définissant deux modèles : un modèle générique pour le problème d'allocation de ressources en ligne avec véhicules autonomes - Autonomous Vehicles - Online Localized Resource Allocation (AV-OLRA)et un modèle multi-agents générique pour les méthodes de résolution de ce problème. Ces modèles permettent la mise en œuvre d'un cadre multi-agents pour comparer les méthodes alternatives de résolution des problèmes ODT. Ensuite, des outils analytiques pourraient être utilisés pour soutenir les décisions de choix entre les méthodes de solution en fonction du contexte spécifique.

Questions de recherche

L'objectif principal de cette thèse est de définir un cadre, c'est-à-dire des directives de mise en œuvre et des méthodes analytiques pour offrir une aide à la décision dans le choix des méthodes de solution ODT dans différents contextes. Nous construisons un tel cadre en nous fondant sur les résultats de l'état de l'art de la recherche sur l'allocation des ressources multi-agents et les problèmes de transport. Nous abordons cinq questions de recherche dérivées de cet objectif global. Les trois premières questions sont axées sur la modélisation des principes fondamentaux de cette thèse ; la quatrième porte sur l'évaluation des méthodes de résolution. La dernière question porte sur les limites de l'implémentation des solutions de l'état de l'art pour aborder l'aspect dynamique du problème.

Question de recherche 1. Quels sont les principaux éléments nécessaires à la formalisation du problème AV-OLRA ?

La première pierre angulaire de notre thèse est la généricité du modèle. Les problèmes ODT peuvent être considérés comme une super-classe de nombreux sous-problèmes et instanciés avec différents scénarios et contextes. Cela soulève le défi de choisir le niveau d'abstraction, ainsi que le choix des éléments qui doivent être inclus dans le modèle abstrait, et ceux qui font partie d'une instance.

Question de recherche 2. Comment pouvons-nous définir une représentation uniforme des différentes méthodes de solution ?

La généricité doit prendre en compte non seulement les instances de problème mais aussi les approches pour concevoir une solution. Les méthodes de résolution des problèmes ODT sont variées. En plus d'une grande variété de mécanismes d'allocation, elles peuvent être fondées sur un répartiteur central, une autonomie individuelle ou des décisions collectives et coopératives, ainsi que différer dans le choix du partage de l'information et des niveaux de coopérativité.
Le défi ici est de définir un modèle multi-agent générique pour les solutions à ce problème. Ce modèle doit s'adapter à différentes méthodes de solution en définissant des niveaux d'abstraction pour l'autonomie, la coopérativité et le mécanisme d'allocation.

Question de recherche 3. Comment pouvons-nous définir une représentation uniforme des

modèles de communication entre véhicules ?

La deuxième pierre angulaire de notre thèse est la connectivité. Les technologies de communication pour véhicules comprennent les équipements, les applications et les systèmes permettant la communication de véhicule à véhicule (V2X), y compris les protocoles DSRC (Dedicated Short Range Communication), LPWAN et C-V2X (Cellular V2X). Les solutions ODT peuvent utiliser soit l'architecture de portail de communication global, soit la communication locale de pair à pair. Le défi consiste ici à définir un modèle de communication capable de s'adapter à toutes les possibilités.

Question de recherche 4. Comment pouvons-nous évaluer la faisabilité et la qualité des méthodes de solution ?

La troisième pierre angulaire de notre thèse consiste à réaliser une évaluation équitable et complète des méthodes de résolution. Une fois que les représentations uniformes mentionnées ci-dessus sont définies, les méthodes de résolution sont classées en catégories. Nous explorons ces catégories et implémentons des exemples d'algorithmes pour chacune d'entre elles. Le défi consiste ici à définir des critères d'évaluation équitables adaptés à toutes les catégories. En particulier, la qualité d'une allocation peut être caractérisée par un ensemble d'indicateurs fonctionnels et techniques. Le calcul de ces indicateurs doit être indépendant du mécanisme d'allocation et permet néanmoins d'évaluer sa faisabilité et ses performances.

Question de recherche 5. Est-il possible de définir une méthode de résolution efficace et réactive pour un cadre dynamique et dont le mécanisme d'allocation permet d'éviter les conflits ?

Dans la solution traditionnelle de l'ODT utilisant l'architecture de portail de communication, les demandes des passagers sont envoyées à un portail. Un opérateur de transport en commun calcule la meilleure allocation de véhicules aux demandes dans le back-end, puis répartit les véhicules pour les desservir. Cela exige que le répartiteur ait une connaissance complète de la demande, de l'offre et de l'environnement au début du calcul, ce qui le rend trop difficile à adapter à la dynamique du problème (voir Défi 2). Cette architecture exige que les véhicules aient un accès continu au portail via une infrastructure de communication globale, ce qui peut provoquer un goulot d'étranglement critique en matière de communication du côté du portail (Défi 3).

On pourrait suggérer l'utilisation d'une méthode de solution décentralisée avec un mécanisme efficace de résolution des conflits, comme les enchères, par le biais de la communication locale. Cela peut permettre d'augmenter la réactivité de la solution. Cependant, cela peut également rendre impossible l'obtention d'une allocation satisfaisante dans les scénarios en ligne, car les informations partagées sont limitées dans l'espace et le temps. (Défi 1).

Les grandes lignes de la thèse

Cette thèse est organisée en quatre parties:

Dans la partie I, nous analysons l'état de l'art à la recherche de modèles et de technologies connexes. Dans le chapitre 1, nous discutons de la formalisation des problèmes de transport liés et nous les classons en mentionnant les travaux connexes dans les domaines de la recherche opérationnelle et des systèmes multi-agents.

Dans la section 2 Nous examinons le paradigme de l'allocation de ressources multi-agents

(MARA) dans le chapitre 2 en détaillant ses composants de modélisation et les modèles de solution existants. Nous positionnons ensuite ce travail par rapport aux domaines de recherche de la modélisation et de la simulation à base d'agents dans le chapitre 3.

La partie II est consacrée à la modélisation du problème et à ses méthodes de résolution. Dans le chapitre 4, nous introduisons le problème (*AV-OLRA*) et proposons un modèle générique pour le formaliser. Nous illustrons les composants du modèle et leurs attributs abstraits et définissons les principaux critères d'évaluation de la qualité de l'allocation par le biais d'un ensemble d'indicateurs fonctionnels et techniques, en plus de définir les formulations des fonctions objectives (d'utilité) que nous considérons dans cette étude. Dans le chapitre 5, nous définissons plus en détail notre modèle générique de programmation orientée multi-agents proposé pour les méthodes de solution *AV-OLRA*. Nous illustrons l'architecture du système multi-agent, en définissant les composants de l'environnement et le comportement abstrait de l'agent. Enfin, nous identifions les fonctions d'utilité et d'objectif des agents que nous considérons dans cette étude.

Dans la partie III, nous identifions les directives générales de mise en œuvre pour chaque catégorie des méthodes de résolution et nous listons quelques exemples d'approches avec des exigences de mise en œuvre détaillées. Le chapitre 6 est consacré au *Dispatching* par répartiteur centralisé et propose un modèle de programmation linéaire adaptatif pour travailler sur des paramètres dynamiques. Dans le chapitre 7, nous présentons la solution décentralisée en détaillant la spécification des approches *Selfish, Cooperative,* et *Market-based*. Puis, dans le chapitre 8, nous décrivons notre approche basée sur le marché (ORNInA) comme une approche efficace, réactive en ligne avec une amélioration continue de la qualité.

Dans la partie IV, nous validons notre proposition de manière expérimentale. Nous implémentons un cadre multi-agents générique pour les problèmes de transport à la demande fondé sur notre proposition de modèle générique. Ce cadre, que nous décrivons dans le chapitre 9, sert de banc d'essai de comparaison par la simulation pour une variété de méthodes de solution. Nous décrivons en détail les méthodes mises en œuvre et les paramètres expérimentaux. Nous effectuons des tests sur des données de scénarios synthétiques et réels. Dans le chapitre 10, la génération du scénario est décrite ainsi que la structure de l'ensemble de données utilisé. Les résultats de l'évaluation sont ensuite discutés et analysés.

Enfin, nous concluons cette thèse par un sommaire de notre travail et soulignons les directions de recherche futures.

Introduction

The first known trial to design driver-less vehicles stretch back to 500 years ago, when motorized vehicles were not even a gleam in someone's eye. Sometime around the year 1478, Leonardo da Vinci drew out his plans for the world's first self-propelled vehicle [57]. His vehicle (cart) was not meant to carry passengers on board. It was capable of moving and changing direction without being pushed. It was powered by coiled springs. It also featured programmable steering, which is achieved by arranging wooden blocks between gears at pre-set locations. Oddly enough, it could only turn right. In the late 1960s, John McCarthy, in an essay at Stanford, imagined that an automated vehicle might one day drive him to the airport [109]. In 1987, Dickmanns and Zapp demonstrated a self-driving Mercedes van in Germany's highway (autobahn); it could stay in lanes and overtake [46]. However, these examples are considered automated, while more than 30 years later, autonomy is not fully achieved yet [134].

Modern vehicles (starting from Level 2 of automated driving defined by SAE[4]) have been equipped with many internal sensors in the last decades: fuel level, engine temperature, battery level, seat belt reminder, etc. These sensors are mostly meant for monitoring the safe state of the vehicle. With the advent of cooperative, connected, and automated mobility (CCAM), external sensors like radars and cameras have been introduced. The purpose of these sensors is to detect the presence and behavior of other transport users. Vehicles with these sensors can collect data (so they "see" and "hear" to build their knowledge models of the surrounding world).
Connectivity information is usually handled by vehicle control and not seen as part of sensor fusion. However, through connectivity, vehicles might share their world models. Over the past decade, both short- and long-range vehicle communication technologies have been developed and introduced in the transport domain with the primary goal of improving traffic safety and efficiency. Vehicle communication technologies comprise equipment, applications, and systems to enable vehicle-to-everything (V2X) communication. These capacities open the path to the cooperative driving application domain. Besides being individual autonomous entities, groups of vehicles (such as those owned by the same company) may behave as fleets and benefit from their collective intelligence, gather information, share it with each other, adapt to the surrounding traffic and environment conditions, to achieve their common goals. In this study we are interested in cooperative driving as enabling technology to tackle the *On-Demand Transport* problem. This problem is also known as *Dial-a-Ride Problem*, in which we have a fleet of vehicles distributed in different locations in the city to serve passenger requests to travel from pick-up to delivery locations respecting a set of request constraints.

[4]SAE stands for the Society of Automotive Engineers, a United States-based, globally active professional association and standards developing organization for engineering professionals in various industries. Originated in the early 1900s as the Society of Automobile Engineers

Motivation

Transport systems are changing rapidly, due to technological developments like Artificial Intelligence and the emerging communication technologies (e.g. cellular 5G, the wireless LP-WAN protocols, and the ad-hoc vehicle-to-vehicle communications via ITS-G5 protocols). Moreover, changing preferences of companies and customers highlight the need for customized On-Demand Transport (ODT) and Smart Mobility Systems. These new mobility solutions may have the capability to change the transportation domain fundamentally. For instance, ODT can lead to the transition from vehicle ownership to vehicle as a service, which would reduce the number of vehicles, as well as increase transportation efficiency and thus reduce unnecessary movements. These potential benefits are claimed to have environmental, social, and economic impacts [54]. The development of autonomous vehicle fleets, capable of peer-to-peer communication, as well as the interest in on-demand solutions (e.g., Uber, Lyft, Heetch) are the primary motivations for this study. We are interested in studying the problem of setting up a fleet of autonomous vehicles capable of responding dynamically in the absence of central control to on-line trip requests throughout a city. The multiagent domain is well suited to the modeling and development of decentralized systems. Therefore, vehicle allocation is a relevant application area for multiagent techniques [49, 100, 137, 143]. On the other hand, centralization of the allocation process with an automatic dispatcher is still quite common in multiagent approaches [49, 143, 102].

Challenge 1. Allocation objective trade-offs:

Allocation problems are major issues in the management of ODT systems. They have been studied for decades, and various solutions have been proposed. In practice, the feasibility and efficiency of the choice to centralize or decentralize the solving methods depend on the problem complexity, its constraints and the environment dynamics. However, even the most straightforward cases of resource allocation lead to debates on efficiency versus fairness [98], efficiency versus privacy [12] and on business quality versus user experience [140].

Challenge 2. Dynamic settings:

The solutions for resource allocation problems in the dynamic settings of ODT systems must challenge vehicle schedules in real-time. This challenge makes the achievement of a globally optimal solution an elusive goal in practice. However, the design of improving approaches for feasible solutions is a suitable alternative to tackle the dynamic aspect issues: this requires taking the communication aspect into account and providing robust and efficient communication and coordination mechanisms.

Challenge 3. Communication bottleneck:

One of the main issues for using multiagent systems (MAS) and multiagent resource allocation (MARA) approaches to solve ODT-related problems is the communication bottleneck. According to Jin and Jie, each agent needs only to communicate with a limited set of neighbors in its planning area instead of communicating with all peers [81]. The development of decentralized solutions based on vehicle-to-vehicle (V2V) interactions could be an essential source of savings and resilience.

Our thesis is that we can address the above challenges by defining two models; a generic model for the Autonomous Vehicles - Online Localized Resource Allocation (AV-OLRA), and a generic multiagent model for solution methods of this problem. These models allow the implementation of a multiagent framework for comparing alternative methods to solve ODT problems. Then, analytical tools could be used for supporting decisions to choose among solution methods depending on the specific context.

Research questions

The overarching objective of this dissertation is to define a framework, i.e. implementation guidelines and analytical methods to offer decision support in the choice of ODT solution methods in different settings. We build such a framework based on state-of-art results from multiagent resource allocation and transportation problems research. We address five research questions derived from this overarching objective. The first three questions are focused on modeling the fundamental principles of this thesis; the fourth is about evaluating solution methods. The last one addresses the limitations in state-of-art solution implementation to tackle the dynamic aspect of the problem.

Research Question 1. What are the main elements required to formalize AV-OLRA problem?

The first cornerstone in our thesis is the model genericity. ODT problems could be seen as a super-class of many sub-problems and instantiated with various scenario settings and contexts. This raises the challenge of choosing the abstraction level, along with the choice of what elements must be included in the abstract model, and which ones are part of an instance.

Research Question 2. How can we define a uniform representation of the different solution methods?

The genericity must consider not only the problem instances but also the solution approaches. Solution methods to ODT problems are various. In addition to a huge variety of allocation mechanisms, they can be based on a central dispatcher, individual autonomy, or collective and cooperative decisions, as well as differ in the choice of information sharing and cooperativeness levels.
The challenge here is to define a generic multiagent model for the solutions to this problem. This model should adapt to different solution methods by defining abstraction levels for autonomy, cooperativeness, and allocation mechanism.

Research Question 3. How can we define a uniform representation of vehicle communication models?

The second cornerstone in our thesis is connectivity. Vehicle communication technologies comprise equipment, applications, and systems to enable vehicle-to-everything (V2X) communication, including Dedicated Short Range Communication (DSRC), LPWAN protocols, and Cellular V2X (C-V2X). ODT solutions can either use the global communication portal architecture or local peer-to-peer communication. The challenge here is to define a communication model capable of scaling to every different alternative.

Research Question 4. How can we assess the feasibility and quality of solution methods?

The third cornerstone of our thesis is to achieve a fair and comprehensive assessment of solution methods. Once the above-mentioned uniform representations are defined, solution methods are classified into categories. We explore these categories and implement example algorithms of each. The challenge here is to define fair evaluation criteria suitable to all categories.
In particular, the quality of an allocation can be characterized by a set of functional and technical indicators. The computation of these indicators should be independent of the allocation mechanism and still helps assessing its feasibility and performance.

Research Question 5. Is it possible to define a solution method that is efficient and responsive for a dynamic setting and whose allocation mechanism allows to avoid conflicts ?

In the traditional solution to ODT using the communication portal architecture, passenger demands are sent to a portal. A transit operator calculates best allocation of vehicles to requests in the back end, then dispatches vehicles to serve them. This requires the dispatcher to have

complete knowledge about demand, supply and the environment at the beginning of the calculation, which makes it too difficult to scale for problem dynamics (see Challenge 2).
This architecture requires the vehicles to have continuous access to the portal via global communication infrastructure, which may cause a critical communication bottleneck on the portal side (Challenge 3).
One could suggest the use of a decentralized solution method with an efficient conflict resolution mechanism such as auctions through local communication. This may lead to increase the solution responsiveness. However, it may also make it unattainable to achieve satisfactory allocation in online scenarios, because shared information are limited in space and time. (Challenge 1).

Dissertation outlines

This dissertation is organized into four pars:

In Part I, we analyze the state-of-art in search of related models and technologies. We discuss in Chapter 1 the formalization of related transport problems and classify them by mentioning related works from both Operations Research and Multiagent domains.
We look at the Multiagent Resource Allocation (MARA) paradigm in Chapter 2 detailing its modeling components and existing solution models. We then position this work with respect to agent-based modeling and simulation research domains in Chapter 3.

Part II is dedicated to modeling the problem and its solution methods. In Chapter 4, we introduce the (*AV-OLRA*) problem and propose a generic model to formalize it. We illustrate the model components and their abstract attributes and define the main allocation quality evaluation criteria via a set of functional and technical indicators, in addition to defining the formulations of objective (utility) functions that we consider in this study.
In Chapter 5, we define in further detail our proposed generic multiagent oriented programming model for *AV-OLRA* solution methods. We Illustrate the multiagent system architecture, defining the environment components and the abstract agent behavior.

We classify solution methods to AV-OLRA into different categories (*Dispatching, Selfish, Cooperative,* and *Market-based*). In Part III, we identify the general implementation guidelines for each category and list some examples of approaches with detailed implementation requirements. Chapter 6 is dedicated to centralized dispatching and proposing an adaptive linear programming model to work on dynamic settings. In Chapter 7 we present the decentralized solution detailing the specification of *Selfish, Cooperative,* and *Market-based* approaches. Then in Chapter 8, we describe our contributed market-based approach (ORNInA) as an efficient, online responsive approach with continuous quality improvement.

In Part IV, we validate our proposal experimentally. We implement a generic multiagent framework for on-demand transport problems based on our generic model proposal. This framework, that we describe in Chapter 9, serves as a simulation-based comparison test bed for a variety of solution methods. We describe in detail the methods we implemented and the experimental settings. We conduct tests on both synthetic and real-world scenario data. In Chapter 10, the scenario generation is described in addition to the used data set structure. The evaluation result is then discussed and analyzed.

Finally, we conclude this dissertation with a summary of our work and highlight future research directions.

Part I

State of the Art

Chapter 1

Background on On-demand Transport Problems

On-Demand Transport (ODT) systems have attracted increasing attention in recent years, particularly concerning road systems. ODT systems have never been considered for the replacement of public transport services but as an extension of them. According to Bellini et al., the concept of ODT was formulated for the first time in the United States around 1990 as a solution to the growing disaffection of potential users, especially at night [14].

Within ODT, allocation problems rise as the most studied optimization problems in the literature. The Autonomous Vehicles - Online Localized Resource Allocation (AV-OLRA) problem concerns assigning, in dynamic settings, transport requests to vehicles under a set of constraints.

Transportation problems can be classified into different categories depending on whether their primary purpose is to minimize the cost, or to maximize the profit of shipping goods, or people.

The problem at hand that we refer to in the rest of this review as Dial-A-Ride Problem (DARP) is know in the literature of Operations Research to belong to the family of Pick-up and Delivery (PD) problems. It can be considered as a special variant of the Vehicle Routing Problem with Pick-up and Delivery and time windows (VRPPDTW) and can be seen as a problem combining scheduling, allocation and routing sub-problems. This chapter presents an overview of the available scientific literature on the principal aspects related to AV-OLRA. We provide an overview of the various vehicle routing problems problems, the specifications of DARP, and their existing solution methods in the literature of the Operations Research.

1.1 Vehicle Routing Problems

Vehicle Routing Problems (VRPs) lie at the heart of distribution management. Thousands of companies and organizations are involved in the delivery and collection of goods or people deal with such problems every day. VRPs are defined through a set of locations to be visited. The distances between all pairs of location are given. In addition to distance values, other values can be used, such as travel time or the cost of the required amount of fuel. A set of vehicle routes such that the total distance, travel time, or total cost, respectively, is minimized, is sought for.

The basic VRP problem is the Capacitated Vehicle Routing Problem (CVRP). Having a fixed

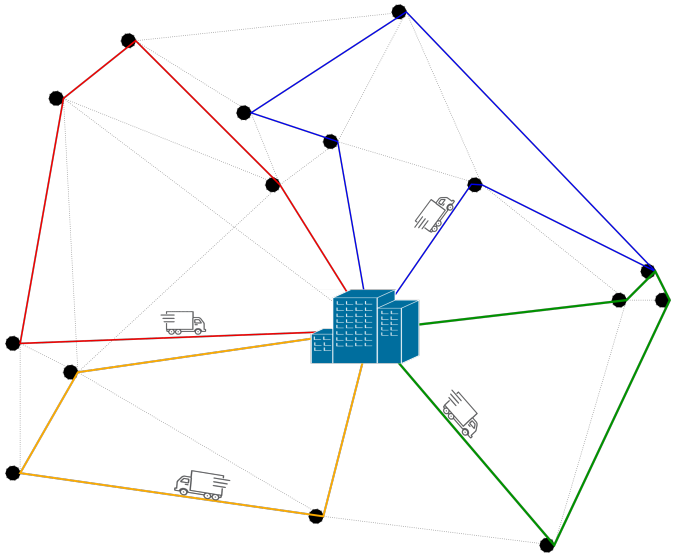


Figure 1.1: The vehicle routing problem

fleet of vehicles with associated vehicle capacity, in the CVRP, we seek for a set of m routes with minimized total cost subject to the following constraints:

- Each route starts and ends at a depot, and is performed by one vehicle
- Each vehicle has a capacity, and performs at most one route
- Each customer has an associated demand, and is visited exactly once by a route
- Customer demands served by a route do not exceed the vehicle capacity Q, which is the same for all vehicles

Moreover, a constraint can be imposed s.t. The length of each route does not exceed the predefined limit (L) that corresponds to vehicle average energy consumption by distance unit and its (fuel/electric) depot capacity.

Vehicle Routing Problems (VRPs) are NP-hard problems as the CVRP (the basic one) generalizes the travelling salesman problem (TSP) which is a special case of the CVRP where $Q = \infty$. NP-hardness implies that no algorithm is known capable of solving each instance in polynomial time w.r.t. the size of the instance itself, and it is believed that no such algorithm exists. However, there is no mathematical proof of this assumption. Efficient algorithms have been found for special variants, or to calculate approximate solutions (heuristic algorithms) with convenient solution-quality/computation-time ratio, so as to provide near optimal solutions for instances with a real application interest [37]. In some of the worst common variations of the (C)VRP, vehicles can differ in their capacity [9], or feature a route length limit to model their autonomy in terms of fuel or electric power (distance constrained VRP [5]), or be allowed to perform more than one route during a day shift VRPs [25], possibly refilling at designated facilities (VRP with Intermediate Replenishment Facilities [61]).

1.1.1 Pick-up and Delivery Problem

In Pick-up and Delivery Problems (PDP), transport operations must be performed from a set of origins to a set of corresponding destinations without transshipment elsewhere [138].

The transport network is modeled as a complete graph $G = (N, E)$ where N is the set of nodes representing geographical locations and E is the set of all edges between nodes. A fleet

of vehicles is available to manage the routes. Each vehicle has a certain capacity, a place of departure, and a place of arrival. Each transport request specifies the load to be transported, as well as its origin and destination, and must be served by exactly one vehicle. On the other hand, each vehicle can serve more than one transport request. A solution to PDP is a feasible pickup and delivery plan minimizing the operational cost. More precisely:

Given a set of vehicles V, and a transport network $G = (N, E)$, a pickup and delivery plan is a set of routes $R := \{R_v | v \in V\}$ such that:

- R_v is a pickup and delivery route for each vehicle $v \in V$.
- Each R_v is a sequence of $n_i \in N_v$ where $\{N_v | v \in V\}$ is a partition of N
- For each vehicle $v \in V$, the capacity of v is respected all along the route R_v

In the work of Savelsbergh and Sol [138], the Generalized Pick-up and Delivery Problem (GPDP) is defined as the general framework for all Pick-up and Delivery problems. In it, n requests must be served, each associated with a set of origins and destinations to be visited, respectively, to pick-up or deliver goods or people. A fleet of m heterogeneous vehicles, each with its own capacity, is used to serve requests. A request is unsplittable (must be served entirely by a single vehicle), and all origins must be visited before any destination, without transshipment in intermediate locations. Each vehicle has specific start and end locations and can serve more than one request, provided that all along its service route, its load is nonnegative and within its capacity. VRP is the particular case of the GPDP with the same capacity for all the vehicles and only one origin or destination per request, the depot, which also acts as the start/end-point of all routes. Another particular case of GPDP is the VRPPD, in which each request has one origin and one destination, and the vehicles are heterogeneous but all based at a depot[36]. Cordeau et al. also call it the n-commodities PDP to distinguish it from the single-commodity and two-commodity PDPs, which can arise in some special cases, for example, in the delivery of beverages in which vehicles deliver full bottles and collect empty ones.

In the same line of work, the Vehicle Routing Problem with Pick-up and Delivery and time windows (VRPPDTW) is tackled, motivated by the fact that most practical VRPPD applications enforce restrictions on time slots in which a vehicle can visit a site. VRPPDTW is defined on a directed graph with $2n + 2$ nodes, i.e., as many nodes as twice the request plus the two nodes needed to be able to represent outgoing and incoming trips of the depot at which vehicles are based. Vehicles are heterogeneous and must complete their routes within a given time horizon, and service at a node must begin within a time window associated with it. VRPPDTW is NP-hard as it generalizes the TSP with Time Windows and Precedence Constraints (TSP-TWPC), which is known to be NP-hard (see e.g. Mingozzi et al. [111]). The 3-index Mixed-Integer Linear Programming (MILP) formulation proposed by Cordeau et al. [36] allows to consider a variant in which vehicles have different start and endpoint, and which is not easier than the basic VRPPDTW version.

1.2 Dial-A-Ride Problems

In the previous section we overviewed the specification of some VRP variants in the literature. Here we dig into the Dial-A-Ride Problems (DARPs) which define a set of variants of VRP dedicated to the transportation of people. In DARP, requests involve transporting passengers. Origins are more often called *pick-up* points, and destinations are called *delivery* or *drop-off* points (see Figure 1.2).

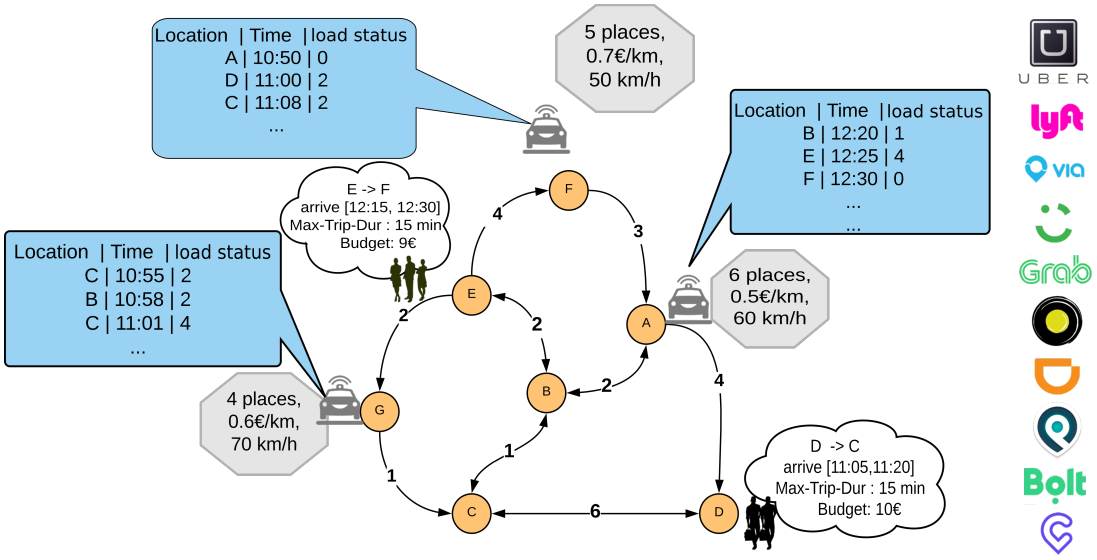


Figure 1.2: The dynamic Dial A Ride Problem with some of its real world applications

1.2.1 Variations of Dial-a-Ride Problem

A common example in many countries is door-to-door transportation for the elderly or disabled persons (i.e. Paratransit [23, 24]). Häll et al. presented an integrated version of the DARP (IDARP), in which some parts of each journey can be carried out by a fixed route public transport service, without synchronizing timetables, which assumes the services are frequent. In many of the practical cases described by IDARP, it is useful and reasonable to allow a request to end its journey on a transfer node near its drop-off node without taking another vehicle for the last leg (part) of the journey [69]. Likewise, there may be cases where it is reasonable, from reducing costs and user-inconvenience points of view, to start the journey on a transfer node rather than on the original pick-up locations. This has led Posada et al. to propose the integrated dial-a-ride problem with timetables (IDARP-TT), in which, under some circumstances, requests may start or end the journey from/to transfer nodes sufficiently close to the pick-up or drop-off nodes without involving the vehicles in the first or last part of the journey. Each request has a given origin, destination, and demand for a set of resources, such as regular seats, wheelchair spaces, and baggage. A request can be served by a single vehicle or transferred between a demand-responsive vehicle and fixed routing system. A heterogeneous fleet of vehicles is located in a depot and used to serve requests; they have different speeds, operating costs, and capacities. The goal is to find vehicle routes that minimize the cost of service on demand and the cost of using the fixed routing system [123]. Bellini et al. propose a classification of DARP based on four operative levels, moving from lower to a higher level provides more flexibility for customers while the related problems get to be more complex [14]:

1. Fixed line service: it is based on fixed routes and fixed stations. Users must reserve the service. Sometimes schedules are also fixed, and buses only make trips if booked; minimal flexibility while scheduling and management is simplest.

2. Fixed routes with the possibility of circumvention: in this case, the paths and schedules are partially fixed; they can be changed at the user's request with potential conversions at certain fixed points, and the entire path integrating the whole itinerary with fixed optional stops. In some cases, this service is called a "corridor service" simply to indicate the limit of the spatial circulation that the trip may be. This level presents a similar configuration to the previous alternative but introduce possible variations of itineraries.

3. Service with free routes within fixed points, where it is possible to specify:

- Zoned service, based on transport routes to fixed public interest points such as parking spaces and railway stations (many-to-few mode).
- Wide service, operating in large areas in general, with complete flexibility over time and free routes at fixed breakpoints (many-to-many mode).

These services range from fixed timetables to on-demand, real-time services.

4. Free route service between unspecified points. This level provides free routes between unspecified stops. Works like taxi service (door to door mode), but it introduces the possibility of variable size of parties and sharing the service with other people along a similar route.

1.2.2 Specifications of Deal-a-Ride Problems

As stated by Cordeau et al., DARP is actually a version of the basic VRPPD in which the objective mostly revolves around Quality of Service (QoS) [36]. The vehicle capacity is typically limited in DARP, while it is often redundant in VRPPD applications (particularly those related to the collection and delivery of letters and small parcels [34]). In some variants of the problem, users associated with separate requests can share the same vehicle as long as its capacity is not exceeded. More specific DARPs allow for one passenger only and thus also restrict vehicles' capacity to 1. Also, a maximum travel time is associated with each request. It corresponds to the maximum travel time between the pick-up point and the delivery point.

The work of Cordeau et al. then reviews the different cases of DARP in the literature, like single and multi vehicle cases, or the different functions to model passenger inconvenience (possibly giving rise to quadratic or convex functions) and how its minimization is dealt with. Quality of Service can replace, or being evaluated along with, some cost related terms, as well as DARP-specific constraints, as e.g. maximum waiting time or maximum excess in user ride times.

For Parragh et al. [119], the notion of time-window is integrated into the basic definition of DARP, as we will consider in the following of the present work. They review some of the most common real-world aspects that are dealt with in literature, especially for what concerns the diversity of vehicles, which can differ not only in capacity but also for what concerns e.g. speed, travel cost, equipment, number of persons that can be transported, transportation mode depending on the passengers, the possibility of accompanying persons. As Cordeau and Laporte pointed out, the primary consideration in some of such a diversity of problems is to tackle the strategic problem of determining size and composition of the fleet that will satisfy the entire demand, while in others, the goal is to maximize the number of trips that can be offered with a fleet of fixed dimensions [34]. A compromise is to serve part of the demand with a basic fleet of vehicles and -if necessary- use additional vehicles (e.g., regular taxis). Parragh et al. also propose a wide review of exact methods for static DARP variants, as well as and heuristic approaches for static, dynamic, and stochastic variants that are found in the literature [119].

Along with static dial-a-ride services, in which all requests for transport are known in advance, there also exist dynamic services, in which requests are progressively revealed throughout the day, or existing requests are canceled, and vehicle routes are adjusted in real-time accordingly, trying to do so without causing too much trouble for other passengers. In the literature, dynamic variants of DARP can be found, although in practice, pure dynamic DARPs rarely exist because a subset of requests is often known in advance, according to Cordeau and Laporte, who report some examples of dynamic DARP. Madsen et al. [106] developed an insertion algorithm, REBUS, based on the ADARTW procedure of Jaw et al. [78] for a real-life problem involving services to elderly and disabled people in Copenhagen [34]. Requests arrive dynamically along a time horizon and are inserted in existing routes considering the difficulty of insertion. The algorithm is reportedly capable of reasonable quality-time compromise solutions on a 300-customer, 24-vehicle instance. Colorni and Righini use three different objectives: the number of serviced

requests, the perceived QoS level by users, and the traveled distance. The insertion of new requests is based on a clustering phase and a routing phase. The routing algorithm applies branch-and-bound to a set of requests that are closer to occur according to time windows [33]. The authors report that they have performed experiments in two cities in northern Italy. Finally, Coslovich et al. used an off-line/on-line two-phase strategy for inserting a new request into an existing route, the objective being minimum user dissatisfaction [38].

To conclude this review of Dial-A-Ride Problems, we cite some more recent works. Gschwind and Drexl define an Adaptive Large Neighborhood Search (ALNS) to find a set of minimum-cost routes on a DARP variant with time windows and additional constraints on and maximum user ride times [66]. Muelas et al. propose a distributed algorithm to solve large scale DARP instances: tests on a set of 24 different scenarios with up to 16,000 requests or 32,000 locations in the city of San Francisco prove it useful [114]. Vallee et al. tackle a dynamic DARP found in a mobility service operated by a private company, in which service requests are either in advance or in real-time and get an immediate answer about being accepted or rejected. The main goal is to maximize the number of accepted requests. Three main on-line reinsertion heuristics (HDR, GH, IGH) based on different neighborhoods are proposed [154]. Bongiovanni et al. present the e-ADARP, a DARP variant which considers the use of autonomous electric vehicles, thus introducing battery management, charging stations, recharge times. The problem's goal is to minimize a weighted objective function of vehicles' total travel time and excess ride-time of the users. They propose a 3-index and a 2-index MILP formulations, along with a branch-and-cut algorithm with new valid inequalities [16]. Brevet et al. address the dial-a-ride problem (DARP) using private vehicles and alternative nodes (DARP-PV-AN), in which to achieve greater flexibility, the on-demand transportation service can be done either by a public fleet or by clients that use their private vehicles. Several pickup/delivery nodes for the transportation requests are considered to address the resulting privacy concerns. A compact MILP model and an Evolutionary Local Search (ELS) algorithm are proposed [19].

Summary

Under a set of constraints, assigning demands to vehicles can be seen as an intersection between resource allocation and constraint optimization problems. There are many variants of such problems extending the VRPs that we overviewed in this chapter highlighting their specifications. The problem described by the illustrative scenario (Figure 1.2) is a variant of DARP, which belongs to the family of Pick-up and Delivery problems and can itself be considered as a particular case of the Vehicle Routing Problem with Pick-up and Delivery (VRPPD) with time windows. This problem can be seen as both a scheduling and allocation problem. The classical modeling approach for such problems is based on the traditional VRP model. Then Linear Programming (LP), and more specifically, MILP (Mixed Integer LP) can be used to solve OR problems, among which, VRP.

Chapter 2

Multiagent Systems and Resource Allocation

The problem of resource allocation across multiple entities is a central concern in both Computer Science and Economics. It is about finding a feasible allocation of a set of resources to a set of consumers in a such way that minimize the cost of the allocation or maximize the cumulative benefits for the consumers. It has interdisciplinary characteristics that make it relevant for different application areas, including industrial production and planning, network routing, traffic management, transportation, and logistics.

A Multiagent System (MAS) is a computerized system composed of multiple interacting intelligent entities, called agents. Usually, a MAS consists of the Agents and their Environment. The Agents of a MAS may be software entities, robots, people or groups of people; a MAS can contain combined human-agent teams.

Features of MASs make them relevant to model complex systems in which autonomous agents are capable of achieving their goals without user intervention. MAS has been used in several areas such as supply chain coordination [161], change order trading [128], equipment management[148], sustainability assessment [153] and recently in smart cities, smart buildings and other Internet of Things (IoT) applications [30, 31].

Multiagent Resource Allocation (MARA) is a domain that studies how to efficiently distribute multiple resources among multiple agents [97], and how agents can affect the allocation [28].

In this chapter, we overview the efforts done in the literature of MAS and MARA and their applicability to the problem of our interest (AV-OLRA). In section 2.1 we describe the main components of a MARA system, then we describe their common properties in Section 2.2. More specifically, Section 2.2.3 emphasizes works that studied how agents, in a distributed way, can optimize a global objective function. We overview their application in resource allocation as Constraint Satisfaction Problems (CSPs) or Constraint Optimization Problems (COPs) in section 2.3. Then we overview in section 2.4 another paradigm of MARA in which the conflict resolution follows market-based protocols.

2.1 Components of Multiagent Resource Allocation

MARA is relevant for a wide range of application domains, Chevaleyre et al. introduced in a survey four of these problem domains, which are: Industrial procurement, Earth observation satellites, Manufacturing systems, and Grid and cloud computing [28], but MARA has also

been applied to many other areas, such as network routing [107], public transport [150], e-commerce [26], social activities [58], and scheduling [95].

The AV-OLRA problem that we are interested in is an ODT problem that can be seen as a MARA problem, where the resources are the trip requests; the agents (the consumers) are the autonomous vehicles; and the allocation must satisfy the constraints of both requests and vehicles. We aim at simulating the behavior of agents for the different solution methods in order to compare their performance. In what follows, we examine the essential components related to the definition of a MARA problem and discuss its main parameters.

2.1.1 Agents

An agent is defined as "a computer system located in a dynamic environment and able to present an autonomous and intelligent behavior to achieve its design objectives" [133].

Traditional object-oriented programming for simulation uses software objects as entities to perform tasks. Objects have no intelligence to react to unexpected events to achieve a goal or a state because they only follow the instruction provider, i.e. human users or other objects. Objects can only change their policy if and when other objects or users provide an update. On the other hand, autonomy is a key feature of agent-based systems. An (autonomous) agent can perform activities with its own motor or intelligence to solve a problem without external interference from human users or other agents. It can respond to different situations and apply alternative strategies to achieve certain goals [79]. Autonomy, however, can lead to conflicts between the interests of different agents. In such cases, one approach is to assign to a special agent the responsibility to coordinate and resolve these contradictions. Other approaches specify conflict resolution protocols to be followed by agents to handle these contradiction in decentralized manner.

Agents have individual beliefs that represent their informational state, in other words the state that the agent owns about the world (including itself and other agents). Beliefs can also include inference rules, allowing the chain to lead to new beliefs. The use of the term "belief" rather than "knowledge" recognizes that what an agent believes is not necessarily true (and may even change in the future).

Agents may or may not have preferences about the resources they receive. This also holds for resources received from other agents: in the case of network connections, for example, the value of a resource decreases if too many users share it. The latter kind of preferences is called externalities [28]. Note that agents may or may not explicitly declare their preferences (e.g., when negotiating with other agents on several options).

So far, to model the behavior of agents in any solution method to AV-OLRA, it is essential to specify the level of autonomy of agents and their preferences for resources in form of a prioritizing function, as well as the communication rules that the agents will follow to avoid or resolve conflicts in form of coordination protocols.

2.1.2 Resources

Generally speaking, resources refer to items required to perform a task, which need to be assigned/allocated to entities in charge of performing it. They can be classified into two types: continuous and discrete.
Continuous resources such as electricity, can be shared simultaneously by multiple agents for a given lapse of time. A *Discrete resource*, like a passenger, is inseparable. Therefore, once such a resource is assigned to an agent, the other agents can no longer use it. Chevaleyre et al. use the term "divisible/indivisible resources" to refer to different types of a resource. Different agents

can receive different fractions of a divisible resource, while in the case of indivisible resources, it may or may not be possible for different agents to share (jointly use) the same resource. Some examples of indivisible but sharable resources are roads [4], and access to network connections as opposed to items of clothing [28]. In the case of non-sharable indivisible resources, an allocation is a partition of the set of resources amongst the agents. For many purposes, task allocation problems can be considered as MARA instances if we consider tasks as associated resources with a cost rather than a benefit [28]. The type of resource is also distinguished by its behavior as a function of time. Resources that do not change their properties during the assignment phase are called static resources, otherwise they are known as non-static (or dynamic). In most cases, resources are not static because, at some point, changes are likely to occur, either by their number or other properties. The type of resource can have a significant impact on the allocation process later [97].

In this work, we consider passenger requests as non-static, non-sharable indivisible resources. A request may change, for example, its status (announced, picked-up, delivered or unsatisfied) at runtime and it cannot be shared between vehicles.

2.1.3 Objective Function

The objective function represents the global objective of the allocation procedure in terms of aggregation of utilities and preferences. Examples are: maximize social welfare for agents, minimize total waiting time for passengers, minimize costs, and so on [97].

The purpose of resource allocation is either to find a feasible allocation of resources to agents, (for example, looking for any assignment of requests to agents with no potential conflict); or to find an optimal allocation. In the latter case, the allocation in question could be optimal for an entity that manage the process from a global point of view (for example, a solution that maximizes the total income of the fleet); or optimal against an adequate aggregation of the preferences of the individual agents in the system (for example, a resource allocation that maximizes the average utility of the agents). Combinations are also possible: the goal may be to find an optimal allocation in a small series of achievable allocations; and what is considered optimal could depend both on the preferences of a central entity and on the aggregation of the individual preferences of the agents. Of course, when the calculation of an optimal allocation is not possible (e.g. due to computation complexity or lack of time), any progress towards the optimum can be considered as a success [28].

2.1.4 Allocation Procedure

The allocation procedure of MARA is the mechanism to distribute the resources in a way that attempts to optimize a defined objective function [94]. Such procedure can be centralized or distributed. In the centralized case, a single entity decides on the final allocation of resources among the agents, eventually after having obtained the agents preferences over alternative allocations. Typical examples are combinatorial auctions, where the central entity is the auctioneer, and the declaration of preferences takes the form of auctions. On the other hand, in purely distributed approaches, allocations appear as a result of a sequence of local negotiation phases [28].
In practice, both centralized and decentralized approaches to MARA have their advantages and disadvantages. An important argument against centralized approaches is that it may be challenging to find an agent capable of assuming the role of central dispatcher or auctioneer (for example, given his computational skills or reliability). Moreover, in a complex environment, much of the knowledge relevant to the resource allocation problem remains local to specific agents and could also be highly dynamic. This makes centralized optimization techniques impractical. The distributed model seems even more natural in practice, when it is impossible to find optimal allocations, an acceptable way to proceed is to take an initial solution and then progressively,

incrementally improve it by small modifications, which are more likely to happen locally instead of being driven centrally. Progressive improvements to the status can be naturally modeled in *Distributed bargaining* or *Distributed Auction* frameworks. Distributed bargaining is a competitive bargaining strategy also known as zero-sum negotiations, it is used as negotiation strategy in which one agent gains only if the other agents loses something [126]. Distributed-auction-based approaches such as Consensus-Based Auction Algorithm (CBAA) and its generalization to multi assignment, Consensus-Based Bundle Algorithm (CBBA) use a market-based decision strategy as the mechanism for decentralized resource selection, and use a consensus routine based on local communication as the conflict resolution mechanism by achieving agreement on the winning bid values [20].

2.2 Properties of Multiagent Resource Allocation

The concepts that we overviewed in the last section represent the essential components of any MARA system. However, MARA solutions can be characterized by their considerations of several properties such as the agent's utility function, social welfare and cooperativeness.

2.2.1 Utility Function

The utility function represents the degree of satisfaction of an agent for a given allocation. Every agent has a utility value expressed as an explicit value or a relation that determines the most satisfactory solution. An allocation procedure attempts to provide agents with alternative resources that match their utilities as much as possible. Having a set of resources R and a set of agents A; let us define X_R as the set of all possible allocations of resources from R to A. Then, the utility function of each agent $a \in A$ takes the following form:

$$u_a : X_R \rightarrow \mathbb{R}^+, a \in A$$

which returns a numeric value corresponding the level of satisfaction of a for each alternative.

2.2.2 Social Welfare

This term refers to social welfare as studied in Economics and Social Choice Theory. Examples include utilitarian social welfare, in which the goal is to maximize the sum of individual public services, and egalitarian social assistance, in which the goal is to maximize the individual welfare of the agent at the worst moment [28].

Social welfare represents the aggregation of the utility functions of all individuals, that could be calculated in different ways [97].

Welfare Engineering refers to the systematic selection of appropriate social welfare orders for a given MARA application (and possibly the design of new orders); and the design of appropriate criteria of rationality and mechanisms of social interaction for negotiating agents, taking into account different notions of social welfare.

2.2.3 Cooperativeness

Stan Franklin proposes a typology of cooperation, positioning it in the context of multiagent systems [47] as shows Figure 2.1. For him, a multiagent system is considered as independent (non-cooperative) if each agent pursues its own tasks independently of the others. On the other

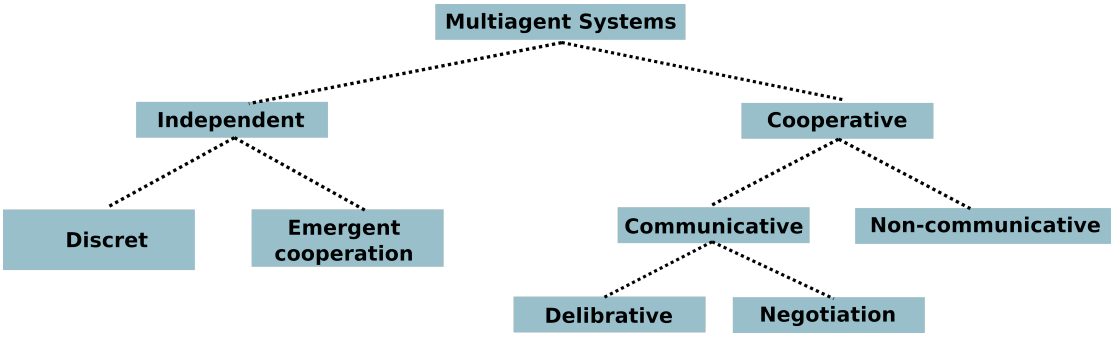


Figure 2.1: Multiagent Cooperativeness typology according to [47]

hand, the cooperation between agents is not necessarily intentional. In this case, from the observer's point of view, the agents appear to be working together, but from the point of view of agents, they are not, as they are simply carrying out their own individual behavior.

Cooperative systems are achieved when the behavior of the agents includes acting together in some way. Such cooperativeness can either be communicative (agents communicate intentionally sending signals to each other in order to carry out their mutual tasks) or it can be non-communicative. In the latter case, the agents coordinate their cooperative activity by observing and reacting to each other's behavior [47].

In order to study how agents, in a distributed way, can optimize a global objective function, Shoham examined four families of techniques that work cooperatively: Distributed Constraint Optimization and Distributed Dynamic Programming; Distributed Markov decision processes; Negotiation; and Coordination on social laws and conventions [144].

On the other side, there exist also non-cooperative techniques that are mostly based on Game Theory, which is defined as a mathematical study to understand situations in which decision-makers interact. A game is meant in the everyday sense as "a competitive activity, in which, players contend with each other according to a set of rules." [77]. In non-cooperative game theory, the basic modeling unit is the individual (including his beliefs, preferences, and possible actions).

A Self-Interested Agent is an agent that has his own description of which states of the world he prefers and that acts in an attempt to induce these states of the world.

Agents will always have utility functions, and they want to maximize the expected utility values. This suggests that acting optimally in an uncertain environment is conceptually simple, at least as long as the results and their probabilities are known to the agent and can be succinctly represented. Agents simply need to choose the action plan that maximizes the expected utility. However, things can become much more complicated when the world contains two or more agents maximizing their utilities with actions that may affect the profits of others.

2.3 Cooperative Resource Allocation

In dynamic MARA, agents can act either competitively or cooperatively. In the field of MAS for distributed problem solving, there is a mathematical framework that combines Artificial Intelligence and Operations Research called Constraint Reasoning (CR) that exploits the homogenization and collaborative interaction of agents [166].

2.3.1 Constraint reasoning

Constraint reasoning techniques help to ensure that the dynamic knowledge that resides in system entities solves the resource allocation optimization problem without imposing unrealistic requirements that all agents continuously communicate their local knowledge to a traditional optimization solver. It is a paradigm in which problems are solved by satisfying the constraints between variables. The notion of a constraint, seen as a restriction on the combinations of values that a set of variables can take, is natural enough to have appeared very early in the history of artificial intelligence.

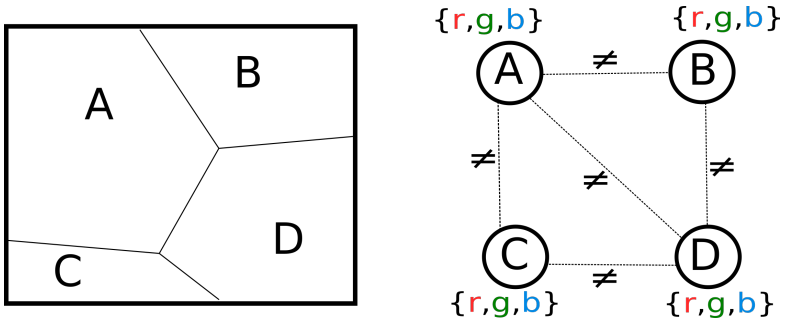


Figure 2.2: Constraint graph example for map coloring problem

A constraint network (a.k.a constraint graph) is a graph whose node set consists of variables, each taking a value in its respective domain, and it's edges represent the constraints that restrict the possible combinations of values between the variables. An example of constraint graph is shown in Figure 2.2 for an instance of the map coloring problem, in this we look for assigning color values (r, g, b) to map regions (A,B,C, and D) in a way that each pair of neighboring regions mast have different colors.

A Constraint Satisfaction Problem (CSP) is the problem of deciding whether a given constraint network has solutions, that is, an assignment of values to all variables that satisfies all constraints. In general, constraint satisfaction is NP-complete, and constraints are generally expressed as binary constraints. Shoham defines CSP by a set of variables, domains for each of the variables, and constraints on the values that the variables could take at the same time. The role of constraint satisfaction algorithms is to assign values to variables consistently with all constraints or to determine that such an assignment does not exist [144].

Constraints can either be *Hard Constraints*, that specify certain requirements for the variables that must be satisfied, or *Soft Constraints*, that have certain values of variables that are penalized in the objective function if, and to the extent that, the requirements on the variables are not satisfied. Many real-world problems are naturally formed as optimization problems where the task is to assign values to variables in such way to maximize utility, minimize cost, etc. while complying with problem-specific constraints. Therefore, CSPs can be extended from satisfaction to optimization by the notion of soft constraints [135].

A Constraint Optimization Problem (COP) is the problem of finding a variable assignment to all variables that satisfies all hard constraints and at the same time optimizes the global cost function. COP is a generalization of CSP in a sense that it allows constraints to be soft associated with a cost of violation, and attempts to minimize the total cost of the solution. Distribution can be considered as a useful extension of classical centralized algorithms to solve CSP where each agent is responsible for assigning one (or several) variables with relative autonomy. Even if it doesn't have a global vision, every agent can communicate with his neighbors in the constraints network. We can then introduce the idea of Distributed Constraint Satisfaction Problem (DCSP) as an idea inspired by classical centralized algorithms to solve CSP and so for Distributed Constraint Optimization Problem (DCOP).

2.3.2 Distributed Constraint Optimization Problem

DCOP is a formalism that captures the advantages and costs of local interactions in a MAS in where each agent chooses a set of individual actions. DCOP is a structure able to model a large number of problems of coordination, programming, and tasks assignment of MAS that is already applied to areas such as sensor networks, scheduling of meetings, and traffic light timing [157]. DCOP has been formalized by Yokoo [166] as a 5-uplet $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \varphi >$ where $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_k\}$ is a set of agents, $< \mathcal{X}, \mathcal{D}, \mathcal{C} >$ is a COP with:

- $\mathcal{X}$: is a finite set of variables, $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n\}$.
- $\mathcal{D}$: is a set of domains, containing a finite and discrete domain for each variable:

$$\mathcal{D} = \{\mathcal{D}_i | i \in \{1, ..., n\}, \mathcal{D}_i \ni \mathcal{X}_i\}$$

- $\mathcal{C}$: is a set of constraints, $\mathcal{C} = \mathcal{C}(R_1), \mathcal{C}(R_2), ..., \mathcal{C}(R_m)$, where each R_i is an ordered subset of variables and each constraint $\mathcal{C}(R_i)$ is a set of tuples indicating mutually coherent values of the variables in R_i.

and $\varphi : \mathcal{X} \rightarrow \mathcal{A}$ is a matching function that partitions variables from $\mathcal{X}$ to agents from $\mathcal{A}$. DCOP aslo has:

- a cost function $f_{ij} : \mathcal{D}_i \times \mathcal{D}_j \rightarrow \mathbb{N}$
- an objective function $F : \mathcal{D} \rightarrow \mathbb{N}$ evaluating an assignment of $\mathcal{A}$ with $F_{ij}(\mathcal{X}_i, \mathcal{X}_j)$ for each pair $\mathcal{X}_i, \mathcal{X}_j$

In distributed algorithms to solve a COP, each agent works in a protocol that combines local processing and communication with its neighbors. A good algorithm ensures that the process ends quickly with a feasible solution (or with the knowledge that there is no feasible solution).

2.3.3 DCOP Algorithms

DCOP algorithms are varied, and the choice between them is dependent on the objective of the solution and the context of the problem. The run-time characteristics of the DCOP algorithm (execution-time, number/size of messages, and memory requirement per agent) is an essential factor when dealing with on-line dynamic problems.

DCOP algorithms can be classified in several ways, for example upon their completeness, run-time characteristics, solution exploration process and synchronization among agents. Fioretto et al. provided a comprehensive overview of the DCOP research area. The survey addresses the different DCOP classifications and applications, including a description of the most typical algorithms of each DCOP family. In terms of the quality of the solution, the algorithms are characterized as either complete (exact), if the optimal solution can be reached, or incomplete, if non-optimal solutions are found (heuristics in which there is a trade-off for reduced computational and communication overhead between the agents). Concerning the constraint information of the agents, DCOP algorithms are partially centralized when agents delegate some constraint information to a central agent, or fully decentralized when each agent has only its own constraint information (e.g., when privacy is a priority issue). In terms of performance of the algorithms, DCOP algorithms are synchronous if the agent's operations and actions are strictly subject to the output of their neighbors, or asynchronous when the agent reacts exclusively based on its perception of the problem.

There are several more classifications such as agent/environment behavior (deterministic or stochastic), algorithm strategy (search, inference...) and complexity measures (time, agent memory, agent communication...), among others [56].

The runtime characteristics of DCOPs identify their memory and communication requirements (e.g., the amount and size of messages) and whether agents only communicate with their neighbors or also with non-neighbor agents.
The resolution (or solution exploration) process of a DCOP algorithm fell into one of the following categories [165]:

- *Search algorithms*: use search techniques to explore the solution space, typically extend centralized search algorithms such as depth-first search.
- *Inference algorithms*: allow agents to exploit the constraint graph structure to aggregate costs from neighborhood. These algorithms are derived from dynamic programming and belief propagation techniques.
- *Sampling algorithms*: rely on sampling the search space to approximate a function such as probability distribution.

According to the way agents update their local information, DCOP algorithms can be categorized as synchronous or asynchronous. Asynchronous DCOPs allow agents to update their allocation based only on their local view of the problem, and thus independently of the actual decisions of other agents. On the other hand, synchronous algorithms force the decisions of the agents to follow a particular order, usually dictated by the implemented representation structure. The implementation details and the specific features of every specific DCOP algorithm is out of the scope of this manuscript. However, taxonomy of DCOP algorithms as presented by Fioretto et al. is shown in Figure 2.3.

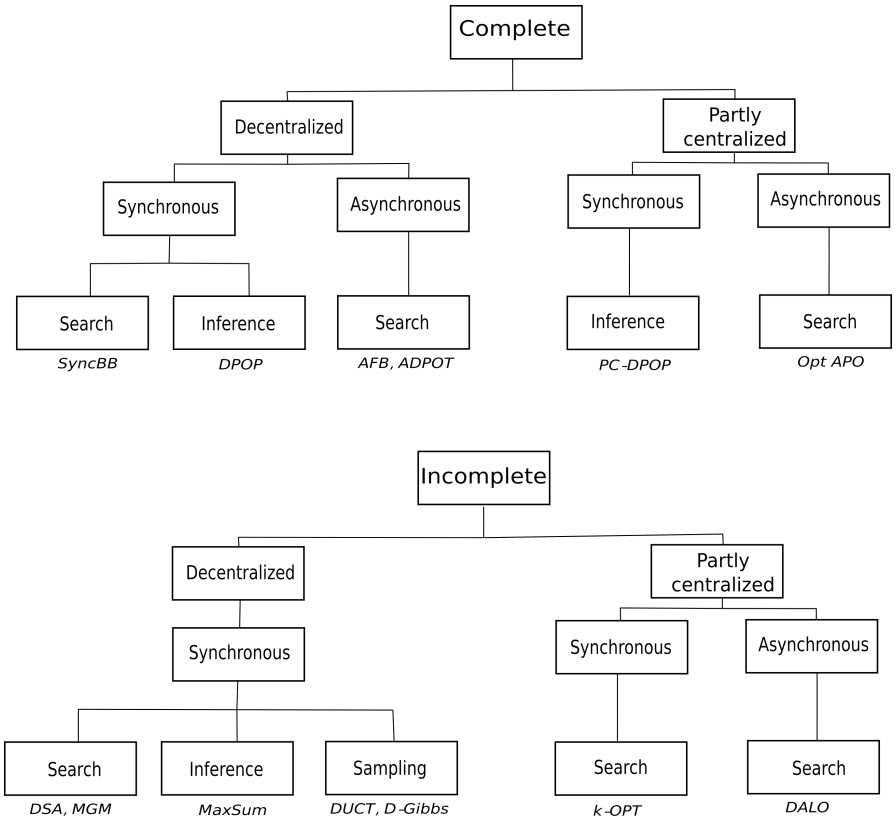


Figure 2.3: DCOP algorithms taxonomy

2.4 Market-based Resource Allocation

The main challenges of resource allocation in distributed systems are critical users acting in their own best interest, and rapidly and unpredictably changing demand. This should be handled efficiently in low latency way. A common solution is a proportional share, where agents each get resources in proportion to their predefined weights. However, this does not allow agents to differentiate the value of their assignments, which leads to economic inefficiency [88].

The function of a market-based resource allocation mechanism is to guide agents in the decisions that determine the flow and consumption of resources. When an agent has to select an action from its available options for each decision, not all actions are possible options at every moment, and some combination may cause conflicts. Thus, the feasibility of an action can be divided into individual feasibility and compatibility. Naturally, an allocation mechanism may guide the agents towards actions that are at least feasible, which can be itself problematic. However, in classical welfare economics, the resource allocation requirements are usually more than feasibility, e.g., attributes such as business efficiency or service quality [76].

Since ancient times, auctions have been used to answer the most fundamental questions in Economy: who should get the goods and at what price? Therefore, auction theory is one of the most significant and widely studied topics in Economics. Some types of auctions are well known, such as the ascending-bid auction or the first-price auction used in many public markets. Auctions are distinguished not only by the terms and conditions of the auction but also by the characteristics of the environment in which they take place, including the number of participants, the number of items traded, the parties preferences, and the form of private information that participants have about their preferences.

A common aspect of auction forms is that information, in the form of *bids*, is received from potential participants about their willingness to pay the price, and the result –i.e., who wins what and how much has to be paid– is determined exclusively based on the information received. This implies that auctions are universal, in the sense that they can be used for many scenarios to determine winners [86], especially in resource allocation [28, 144]. A great deal of coordination approaches have elements that are both centralized and distributed and therefore fit in the middle of the spectrum. Dias et al. position market-based systems under this hybrid category. Market mechanisms can retain the benefits of distributed approaches, including robustness, flexibility and speed. Auctions quickly and concisely gather the information of participants in one place to make decisions about resource allocation; in some cases, they provide solution quality guarantees. Market-based approaches may also incorporate methods for opportunistically coordinating sub-teams in a centralized manner. However, market-based approaches have some weaknesses. In domains where fully centralized approaches are feasible, market-based approaches may be needlessly complex in design and have greater communication and computational requirements [45].

Summary

Allocation problems are central concern in managing ODT systems. Solving resource allocation problems in dynamic environments, such as ODT systems, must challenge the allocation in real-time. This challenge makes it an elusive goal in practice to achieve an optimal solution. However, the design of approaches that start from feasible solutions and then improve them is a suitable alternative to tackle the dynamic aspect issues; this requires taking the communication aspect into account and providing robust and efficient communication and coordination mechanisms. MARA solutions are identified by the behaviors of individual agents and their coordination mechanisms so that various solution schemes exist. We can define three dimensions for the features of these mechanisms:

- The level of decision *autonomy* of agents, that identify whether they take fully decentralized autonomous decisions or they follow some centralized instructions.
- The level of *cooperativeness* of agents, that defines their intention to share information about their decisions or not.
- The *allocation method* adopted by the individual agents, that define how an agent prioritizes a particular resource and insert int into its potential schedule

We have overviewed in this chapter several MARA approaches that we can classify based on their consideration of these dimensions as follows:

- *autonomy* can take value from the set {`centralized, decentralized, partially-centralized`}
- *cooperativeness* value is either `sharing` or `no-sharing` of decision information
- *allocation method*s are varied, some examples are Auctions, DCOP algorithms, Linear programming solutions, etc.

Chapter 3

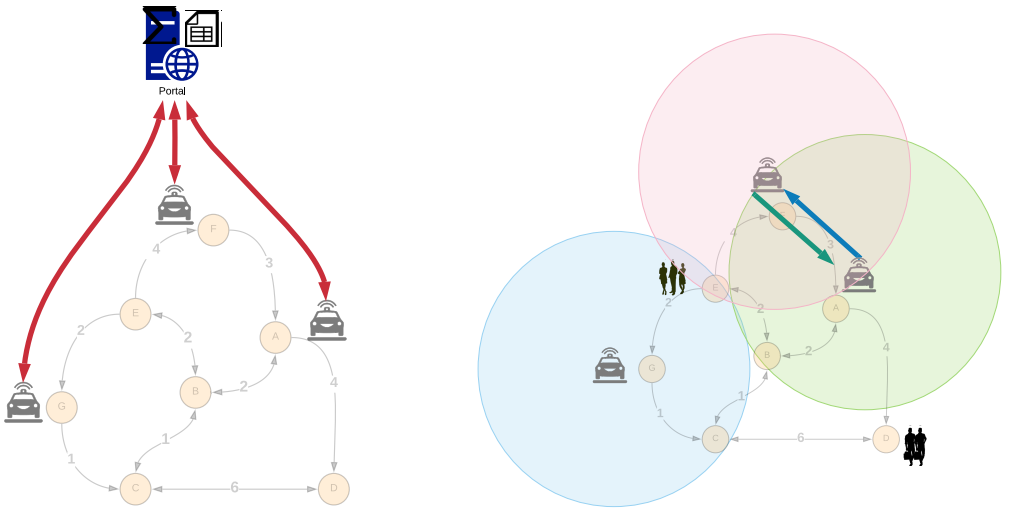
Applying Multiagent Resource Allocation to DARP

In recent years, the number of articles devoted to applying agent-based technologies to transport and communications engineering has increased significantly. Bazzan and Klügl review the existing literature on agent-based modeling and simulation of transport. They stated that agent-based approaches are well suited to traffic and transport management, given the geographic, functional, and temporal distribution of data and control, as well as the frequent and flexible interactions between participants and their environment. Ronald et al. investigate the application of agent-based simulation for studying ODT systems. They identified that existing works and applications are strongly focused on optimizing trips, usually in favor of the operator, and rarely consider individual preferences and needs of customers. In order to make better use of existing transport infrastructure, ODT systems are gaining ground internationally. However, many systems fail due to poor implementation, planning, or marketing, and thus, being able to realistically assess the viability of a system in particular cases is essential.

In this chapter we investigate the application of MARA to an ODT system defined as in the DARP which we introduced in Section 1.2. We first provide an overview of existing MARA approaches to solve DARP, then we define the set of criteria to be used for evaluating the existing work in the literature. After that, we overview the most relevant contributions to solve DARP and the common modeling and simulation frameworks that are used for evaluating such contribution. Finally in section 3.4, we address some limitation of the state-of-the-art models.

3.1 Existing MARA solutions for DARP

Traditional approaches for ODT consider a centralized dispatcher architecture like in Egan and Jakob [49], Shen and Lopes [143] or a decentralized Multiagent System (MAS) to reduce the problem complexity to be handled by a central coordinator like in El Falou et al. [53], Grau et al. [65]. There exist several approaches for decentralized decisions and self-coordination, like Glaschenko et al. [62] which introduce a multiagent bid-based real-time scheduling solution in fully decentralized settings. Here each vehicle represented by an agent can negotiate via radio channels with flexible decision criteria. A pattern recognition algorithm is used to predict the most likely locations for the next demand using agent-based data mining to recommend movements to these locations. Investigating the applicability of Genetic Programming (GP) for developing decentralized MAS that solve dynamic DARP, van Lon et al. present a method to



(a) Central dispatcher (global communication) (b) Decentralized decisions (local communications)

Figure 3.1: Centralized vs. Decentralized architectures

automatically generate a MAS that can solve the DARP for a specific set of scenarios. GP is used to generate a heuristic that is effective in solving the DARP compared to centralized solutions. This approach's best result is by planning only one demand in advance by vehicle, which maximizes the agent's local interests (greedy) and produces a feasible solution very fast [155].

Grau and Romeu propose an agent-based model based on simulated events for the real taxi market, where supply and demand matching depends on event-based interactions. According to their conclusion, one of the main limitations is the assumption of uniform distribution of demand in the service area [64]. Agatz et al. aimed at addressing the uncertainty caused by the dynamic nature of online demands. With ignoring the time required to execute a planning algorithm, they proposed the use of a deterministic *rolling-horizon* solution approach. Plans in the rolling-horizon approach are drawn up using all known information in a planning horizon, that is "rolled" forward to include more available information [2]. Based on vehicle coordination via message passing, and extending the generic model for Online Localized Resource Allocation (OLRA) [167], using P2P communication, the work of Picard et al. propose the Online Localized with Communication Constraint Resource Allocation (OLC^{2}RA) for concurrently solving the allocation problem over a fleet of autonomous taxis, in which a vehicle decide its next destination (scheduling only one demand in advance) [122]. On the contrary, ALMA decentralized heuristic proposed by Danassis et al. is wholly decoupled and does not require direct communication between the participants. They demonstrate an upper bound of the speed of convergence which is polynomial to the desired quantity of resources and competing agents per resource; in the realistic case where the mentioned quantities are limited whatever the total number of agents/resources, the convergence time remains constant as the total size of the problem increases [40]. However, conflict detection still requires communication with other vehicles, resources, or a central entity to enable resources to share information about their status, such as the blackboard coordination mechanism.

3.2 Evaluation Criteria

In the following, we define evaluation/identification criteria to classify existing works accordingly. To assess contributions related to DARP and its agent-based modeling, allocation procedures, solution results, and simulation framework implementation, we define in what follows seventeen criteria classified by their evaluation purpose into five categories: Problem Model, Agency, Procedure, Solutions, and Implementation.

3.2.1 Problem Model

The problem at hand could be dynamic in several aspects. These criteria define how flexible and generic the model is to deal with the dynamic aspect of the problem.

EC 1. Demand model : this criterion concerns the demand parameters considered in the reviewed work, in addition to the nature of the demand distribution, is it statically known in advance or dynamic, is it deterministic or stochastic ...

EC 2. Assessment scenario generation: this criterion is related to the assessment of the proposed models: whether it relies on synthetic data simulation or on real world data sets.

EC 3. Fleet's heterogeneity: This criterion defines if the model supports scenarios with heterogeneous fleets or not.

EC 4. Environment model: this criterion is about the interface between system entities and their environment, Defining such an interface is not obvious. An essential aspect is to respect their autonomy and to ensure the application of environmental rules. Dynamic environments include endogenous processes that allow the state of the environment to dynamically evolve outside the control of the system. In a static environment, this process is not included.

3.2.2 Agency

These criteria describe the general characteristics of the agent models and their abstract system architecture including:

EC 5. Decision autonomy: this defines if agent decisions are taken autonomously or they follow centralized or hierarchical orders.

EC 6. Agent cooperativeness: this defines the level of information sharing between agents and how does this affect their individual decisions.

EC 7. Allocation mechanism: this defines the method to resource allocation e.g. based on combinatorial optimization, constraint optimisation problem, market based auctions, etc.

3.2.3 Procedure

The purpose of these criteria is to describe the features of the allocation procedures to assess their feasibility and performance.

EC 8. Optimality: An *exact method* always finds optimal solutions and yields an optimality proof, although it can be computationally challenging. An *approximation method* finds a sub-optimal solution offering a guarantee of approximation on the quality of the solution found. A *heuristic approach* finds a non-optimal solution whose quality can only be verified experimentally. However, it is sometimes best to deal with the problem and solve it heuristically, especially in the presence of large-scale problems. EC 8 identifies if the reviewed work proposes exact, approximation or heuristic approach to solve the problem.

EC 9. Objective function: this criterion allows to classify the reviewed works depending on the parameters they considered in their objective function.

EC 10. Communication-wise robustness: this criterion classifies the systems based on how they will deal with communication issues such as message loss, message non-conformity, and communication range limitation. This criterion address system robustness by comparing its

performance when confronted to a variety of communication issues cases.

EC 11. Information-wise robustness : This criterion addresses the system robustness against the frequent update in information-related dynamic changes in environment, schedules, demand, and status of system entities.

3.2.4 Solutions

These evaluation criteria concern indicators and metrics for operational assessment of solution methods on specific problem instances

EC 12. Quality of Business (QoB) indicators: these indicators can be seen as perceptual, conditional and subjective attributes, and can be interpreted differently by different people. Consumers may focus on the quality of a product/service's specifications or comparison with market competitors, while service providers could measure the QoB as degree of compliance w.r.t. the manufacture specifications, or by the revenues achieved by reducing expenses and increasing profits.

EC 13. Quality of Service (QoS) indicators: this criterion is influenced by human factors identified by set of indicators including: stability of service quality, service availability, wait times, and user information. Here we focus on how the authors answered to QoS aspects and what are the parameters they considered in order to achieve QoS.

EC 14. Technical indicators: this criterion defines the technical factors of the proposed solution influenced by reliability, scalability and efficiency. Technical indicators include among others: solution computing costs, communication costs, and ease of maintenance.

3.2.5 Implementation

Here we focus on the implementation of the proposed model, simulation framework, and their limitations.

EC 15. Simulation goals: The purpose of this criterion is to identify the main focus of the simulation scenarios and the supported evaluation metrics.

EC 16. Framework genericity: we are interested in identifying whether the contribution build the model as an ad-hoc implementation specific to ODT or develop it on top of a generic framework

EC 17. Transport problem specification: the purpose of this criterion is to specify if the transport characteristics must be coded or is included in the proposed framework, and for those if the ODT specifications (basic vehicle behaviors, solution strategies) are or must be implemented, and so for the communication management and constraints.

3.3 Agent-based Modeling and Simulation

Agent-based approaches can contribute to design and control intelligent transport systems [10] because of their following characteristics:

- they can solve natural and intuitive problems involving active entities with a (potential) local perspective, that are harder to model with than complex centralized entities whose knowledge cannot include all the details and restrictions necessary, because it would be

too burdensome. It can provide adaptive and powerful services thanks to its ability to self-organize;

- autonomous agents provide an appropriate basis for modeling heterogeneous systems. Each entity can have its own structure, representation, and behavior. Therefore, a particular level of detail can be included in a simulation model or in a problem-solving structure that is arbitrarily developed and applied at the agent level;

- high-level abstractions can be used to describe agents and interact with them. Therefore, they provide an intuitive level of interaction between human users and the agent-based system;

- MASs allow to manage the evolution in the system structure elegantly and efficiently. Agents can control the amount of time in which they interact. These (dynamic) relationships can be controlled, separated, or established from a local perspective;

- agents can eventually adapt their behavior to achieve an evolving self-organization capacity. This flexibility can be very useful in the transport and communications sectors.

It is generally accepted that the construction of analytical models (equational) is difficult. This motivated the development of agent-based modeling, which allows to model complex systems simply by involving much simpler modeling of the behavior and interactions of the agents that make up the system.

3.3.1 Evaluating Solution Methods via Transport Simulation

The recent development of Intelligent Transportation Systems (ITS) has led to a growing attention on improving the efficiency and reliability of transit systems. This attention has led to the need for appropriate methodologies and tools to assess ITS efficiency. To this end, a suitable evaluation tool is simulation. The use of simulation is intended as an experimentally-based method to design and create a model of transportation system for use in numerical experiments. The goal is to understand better the behavior of such a system under a given set of requirements [42]. In recent years, agent-based transport simulation approaches have been proven able to acquire the necessary level details to reproduce realistic phenomena. Agents can represent drivers, vehicles or other transport entities. They are explicitly present as active and heterogeneous entities in an environment that represents the road network where they can present complex elaborations of information and perform arbitrary decision-making. Their behavior can be viewed, monitored and validated individually, opening up to new possibilities for analysis, development and illustration of traffic phenomena. The main difficulties of these approaches are computing power limitations, computer memory and (perhaps the most critical) data availability [10].

In the literature, vehicle mobility models are generally classified as microscopic or macroscopic. Microscopic mobility models focus on the mobility of each individual vehicle, while Macroscopic approaches focus on the complete road flow taking into account the general traffic density, vehicles distributions, and various constraints (as crossroads and traffic lights) [71].

The basic structure of microscopic models, such as the Dynamic Route Assignment Combining User Learning and micro-simulation (DRACULA), developed at the Institute for Transport Studies of the University of Leeds (UK) by Liu et al. [96], takes into account two concepts of fundamental importance. The first focuses on the travel choices made by an individual to make a trip: travel objectives, travel needs, perceptions, behavior, and cognitive abilities that influence choice processes derived from the state of these variables when the choice is made. The second, consists in modeling how the network status changes over time, and in considering how the spatial knowledge of a driver changes constantly depending on the movements performed on the entire network.

Rossetti et al. [132] described an extension of the DRACULA traffic simulator to model drivers decision-making without information, with information provided just before the start of the trip, or with information provided both before and during the trip. So that agents update their belief base and select plans (routes) and departure time. This extension combines demand generation, departure-time choice, route choice and micro-simulation of environment with a formal agent-based model for driver's cognitive decision-making, using BDI-based [1] agent specifications Agentspeak language.

3.3.2 Common Simulation Tools

TaxiSim is one of the most common simulation platforms for taxi fleets. Cheng and Nguyen [27] introduced TaxiSim as a multiagent simulation platform, to simulate the operation of taxi fleets. TaxiSim is designed to model individual driver strategies at the micro level. It is also designed to be scalable so as to simulate thousands of vehicles at the same time. Actual operating data, if available, can also be imported into TaxiSim, allowing to create an extremely realistic simulation environment.

To study cooperation behavior of vehicles in order to optimize a flexible transport service, Lammoglia et al. [89] developed a model of a theoretical transport system in the open source multiagent programmable modeling environment Netlogo [116], that compares different optimization processes for two types of services. The first service is driven by stop location attractiveness (selfish vehicles) and the second service is a simple process of communication and cooperation between vehicles (cooperative vehicles). The main objective is to understand if the cooperation between vehicles have a significant effect on the efficiency of the global service. For this study, the authors simulated the combination of the two models to be analyzed, i.e. competition against cooperation. Although it is not surprising that cooperation improves transport efficiency, these experiments show that statistical deviations are important between the two evaluated services. The results of [89] open questions about the integration of competing services. One can clearly believe that any cooperation between different methods can be of importance. This does not seem to be the case when cooperative and selfish vehicles are connected to the same service. In future work, it will be interesting to analyze the sensitivity and relationship between these transport systems: how can one make them cooperate and maintain a good balance between conflicting goals on many levels (taxis, taxi fleets, local transport authority, carrier, etc)? should different parameters be used for different systems which partially overlap or divide only space?

The simulation must incorporate a realistic dynamic of customer demand, traffic flow, and fleet management operations. These aspects are even more important when considering urban areas due to the stronger dynamics of traffic flows, with a consequent continuous evolution of travel time. Using microscopic traffic simulators, especially when tackling dispatching problems, evaluates the performance of the service in different scenarios, often extreme, such as sport/cultural events, bad weather or transport strikes. To keep track of the vehicles, it is necessary to simulate them individually. For the same reason, requests must be microscopic [103]. Maciejewski and Nagel [103], Maciejewski [100], Maciejewski and Bischoff [101] describe how the open source MATSim-simulator [108] can be used to generate problematic instances for a dynamic VRP (DVRP). This is done by coupling a DVRP optimizer to MATSim. Maciejewski et al. [105] designed a Dynamic extension of the Vehicle Routing Problem to MATSim to be highly general and customizable to simulate a wide range of dynamic VRPs.
The extension allows to link several algorithms that are responsible for the continuous re-optimization of the routes in response to changes in the system. The DVRP extension has been used in many research and commercial projects related to the simulation of electric and autonomous taxis, Receptive transportation on demand, fast personal transportation, free carpooling and parking search.

[1] Belief-Desire-Intention (BDI): a software model in agent programming provides a mechanism for separating the plan selection activity from the currently running active plans.

The performance of ODT services with MARA and MAS coordination depends largely on two key factors: (1) the transport control mechanism used for resource allocation and scheduling; (2) the parameters of the implementation scenario, in particular the topology of the underlying road network and the spatio-temporal structure of passenger demand. Understanding how these factors affect the performance of the transport system is essential for the development and implementation of ODT services. For this reason, Čertický et al. [171] and Čertický [170] proposed an open-source agent-based simulation test bed that allows to evaluate the performance of multiagent ODT schemes and to compare them: centralized and decentralized solutions, static and dynamic passenger allocation, as well as vehicle routing mechanisms under various conditions. The test-bed is built on top of the transport simulation framework AgentPolis [3], which is a fully agent-based simulator in order to model transportation systems. Individual entities in the transport system are represented as autonomous agents with continuous asynchronous controllers capable of interacting with their environment and other agents. This allows modeling scenarios where agents modify their plans at any time of the day based on their observations about the environment and/or communicate with other agents.

The work of Cich et al. [29] aimed at evaluating the profitability of precision flow service providers in several years of activity in a specific public compensation condition. Focusing on ODT, Cich et al. proposed a co-simulation model and a framework for simulating supply and demand, particularly for instances with low density flows of trip demands. The cooperative multiagent model which is implemented using the General-purpose Agent-Oriented Programming Language SARL is responsible for providing negotiations between agents, while the MATSim agent-based simulator is responsible for travel planning.

3.4 Analysis of Multiagent Contributions to DARP

It is worthy to mention that the above mentioned models vary in the flexibility degree regarding the characteristics of *demand model* (EC 1), *assessment scenarios* (EC 2), *fleet heterogeneity* (EC 3) and *environment dynamics* (EC 4). The variety of works in ODT literature relying on these platforms also differ in their consideration of these characteristics. Table 3.1 list the different characteristics for the flexibility criteria, from which the choice of any hypothetical combination is possible to define simulation scenario attributes.

EC 1 (Demand model)	**EC 2 (Assessment scenarios)**	**EC 3 (Fleet heterogeneity)**	**EC 4 (Environment dynamics)**
Announcement (on-line/off-line)	Scenario data (synthetic/realistic)	Structural (capacity, speed, etc)	Spatio-temporal (distances)
Distribution (uniform/deterministic/stochastic)	Scenario generation (Parametric/Statistical)	Behavioral (strategy, preferences, etc)	Traffic (artifacts, flow, etc.)
			Communication model
			Observability

Table 3.1: Flexibility criteria attributes considered by modeling and simulation approaches

The variety of conceptual and experimental models for the problem at hand and its solution methods provides a rich, fundamental background for researchers to develop and assess their contributed approaches to solve ODT instances. However, to the best of our knowledge, the literature of ODT and MARA lacks for the definition of guidelines for implementing analytical tools to assess these approaches. Using uniform representations of problem instances and their solutions should allow the implementation of a generic multiagent framework for comparing alternative methods to ODT problem variants. Then, analytical tools could be used for supporting decisions among different solution methods in different contexts.

The dynamic nature of the problem rises the challenge of uncertainty, this leads to the absence in practice of feasible methods which guarantee to achieve optimal solutions. Therefore, works proposing MARA approaches to solve ODT instances focus either on achieving feasible solutions

that are robust against problem dynamics and communication issues, or on reaching good results regarding quality criteria by letting the challenges related to the dynamic aspect of the problem aside.

In Section 3.1, we mentioned that there exist several approaches for decentralized decisions and self-coordination approaches to solve DARP, like Glaschenko et al. [62], van Lon et al. [155], Seow and Lee [142]. Many surveys overviewed these works like Bazzan and Klügl [10], Parragh et al. [118], Desjardins et al. [43]. In what follows, based on these surveys and the evaluation criteria (EC 5. to EC 14.) defined in the previous section, we assess the works in the literature of MAS to solve DARP.

Recalling EC 5 (*decision autonomy*), on the one hand, traditional approaches for ODT in literature consider the *centralized dispatcher* architecture. Some multiagent approaches keep this architecture like in Maciejewski et al. [104], Egan and Jakob [49], Shen and Lopes [143], Gacias and Meunier [59], Kümmel et al. [87]. Some others propose decentralized MAS to reduce problem complexity but still using this architecture in the sake of conflict management using a *central coordinator* like in El Falou et al. [53], Grau et al. [65].
Considering EC 11 (*Information-wise robustness*) it is impractical to find optimal allocations. However, the mentioned works considered that *heuristics* with continuous improvements over an initial feasible allocation of resources would be a success (EC 8).
The cooperation (EC 6) in these architectures depend on the global communication of information with the central entity, which is supposed to be trust-worthy in handle the conflict management and achieve feasible solutions. One of the main issues here is the communication bottleneck, and the robustness against communication issues (EC 10). To the best of our knowledge, none of the mentioned works focuses on robustness against these issues.
The most considerable difference between these approaches is the adopted allocation mechanism. The feasible solution in these approaches are achieved by applying a variety of straightforward allocation methods (EC 7), Alshamsi et al. [6], Shen and Lopes [143], Maciejewski et al. [104] used *greedy* methods with different priority functions, Gacias and Meunier [59] used the *insertion* heuristic (see Solomon [147]), while Kümmel et al. [87] proposed an adapted version of the Gale–Shapley algorithm for stable matching (see Gale and Shapley [60]).

On the other hand, the *decentralized* approaches can differ to a greater extent than allocation mechanisms. Table 3.2 classifies some example contributions based on the level of agent cooperativeness (EC 6) into three categories: global, limited-range, and no information sharing. Unlike the ordinary models where vehicles are considered as autonomous agents, both Bazzan et al. [11] and Egan and Jakob [49] considered the passengers to be the agents who take decisions and negotiate with the supply entities.

Cooperativeness level	**Contributions**
No information sharing	van Lon et al. [155], Egan and Jakob [49], Hrnčíř et al. [75]
Sharing within limited-range	Winter and Nittel [159], Glaschenko et al. [62], Seow and Lee [142], Jin and Jie [81],
Global sharing	Lammoglia et al. [89], Bazzan et al. [11], El Falou et al. [53], Kalina et al. [83]

Table 3.2: Classification of decentralized approaches based on EC 6.

The allocation mechanisms that are adopted by the reviewed works vary considerably, although market-based mechanisms (*Negotiation* and *Auctions* protocols) appear in about 50% of the papers (see Figure 3.2). Other common allocation mechanisms depend on straightforward methods like greedy, insertion and local search heuristics. To ensure optimal allocation, complete decentralized algorithms can be used. For example, El Falou et al. [53] introduced

Figure 3.2: Usage of different allocation mechanisms (EC 7)

an adapted $A*$ algorithm requiring global communication, but the robustness of this algorithm was not assessed regarding the problem dynamics. However, heuristic approaches are the most common in the literature.

The allocation objective function (EC 9) describes the overall potential of the solution from the point of view of either the passengers or the service provider. The objective of the service provider is usually to minimize costs and maximize the profits: works focused on its point of view defined their allocation objectives to maximize the QoB. Passengers usually want to be served with the lowest waiting time and less transfers. Works aimed at increasing passengers' satisfaction focus on maximizing QoS. However, some works combined measures from both point of views for multi objective allocation. Table 3.3 list the most common solution indicators for QoB (EC 12), QoS (EC13) and technical (EC 14) criteria.

QoB indicators	**QoS indicators**	**Technical indicators**
Fleet size	Satisfied requests rate	Computation time
Relative gain (profit)	Waiting time	Optimization rate
Travel distance	nb waiting requests	Priority function choice
Time operational cost	nb transitions	Connectivity
Vacant distance	Response time	Search depth
Idle time	Service stability	nb egotiation rounds
Occupancy rate		nb Messages
Vacant rate		Charging (rate/time)

Table 3.3: Quality and technical indicators (EC 12, EC 13 and EC 14) of solution approaches

Works that propose zone-based communication are Kalina et al. [83], Alshamsi et al. [6], while flexible communication models are proposed by e.g. Jin and Jie [81], Winter and Nittel [159] to form efficient infrastructure, avoid bottlenecks and scale for handling connection issues. On the other hand, no sufficient information to answer to (EC 10) is found in the works based on global communication and shared memory such as Lammoglia et al. [89], Maciejewski et al. [104].

Similar to *centralized* approaches, decentralized approaches that consider the dynamic aspect of the problem (Jin and Jie [81], Lammoglia et al. [89], Egan and Jakob [49]) apply continuous optimization or frequent rescheduling to improve the solution quality and avoid issues related to the uncertainty about demand and the environment.

Recalling the implementation criteria EC 15, EC 16 and EC 17, we notice that advances in multiagent technology have motivated researchers to build microscopic simulation frameworks either as ad-hoc implementations (Deflorio et al. [42], Furtado et al. [58], Hampshire and Sinha [70]) or using generic platforms (e.g. Repast Simphony [117], Netlogo [116]) in which transport characteristics must be coded.
On the other hand, there are MAS platforms dedicated to transport problems, often mainly focused on traffic (e.g. Sumo [13], Movsim [151], Matsim [8], MITSimLab [15]).
The initial goals of these platforms are to simulate traffic flows and demonstrate how traffic states may evolve over time. These models identify and track the behavior of individual vehicles (often referred to as car-following models). To assess the performance of fleet management and control mechanisms, ODT specifications (strategy, basic vehicle behavior, etc.) must be implemented on top of these platforms.

Some notable open source multiagent ODT simulators exist such as RinSim [155], Agents4ITS Mobility testbed [170], and TaxiSim [27] in which communication management, communication constraints and different solution models require to be coded.

Summary

The modeling and development of decentralized systems are well suited to the multiagent domain. Therefore, vehicle allocation is a relevant application area for such techniques. On the other hand, centralization of the allocation process with an automatic dispatcher is still quite common in multiagent approaches. In this chapter we provided a literature review on applying MARA approaches to ODT applications. We first defined the set of evaluation criteria that we used in this review, then depending on these criteria we conducted an analysis on the works that proposed solution methods. The solutions for DARP as a resource allocation problem in ODT systems within dynamic environments must challenge schedules of vehicles in real-time. This challenge makes the achievement of an optimal solution in practice an elusive goal. However, designing improving approaches for feasible solutions is a suitable alternative to tackle the dynamic aspect issues; this requires taking the communication aspect into account and providing robust and efficient communication and coordination mechanisms. Solution methods vary on mutable dimensions, so that a uniform representation and categorisation of different solution models is necessary in order to compare their performance in different context fairly and efficiently. We also overviewed the main notable works in literature that propose modeling and simulation tools. The variety in consideration of problem characteristics express the need for a uniform, scalable representation of problem instances.

To the best of our knowledge, such representations specific to DARP are missing in the literature of MARA and ODT.

Part II

Modeling (AV-OLRA)

Chapter 4

Modeling the Autonomous Vehicle Fleets Allocation Problem

In On-Demand Transport (ODT) systems, allocation problems consist of finding feasible allocations of requests to vehicles, with respect to some domain-specific constraints and objectives. Being autonomous, the vehicles of a taxi fleet can be responsible for their choice of allocation to requests (making *decentralized decisions*), or follow the schedules that are *centrally decided by a dispatcher*. In practice, the feasibility and efficiency of the choice to centralize/decentralize the solution depend on the problem complexity, its constraints and the environmental dynamics.

For a better understanding of the Multiagent Resource Allocation (MARA) for ODT problem, we provide an illustrative scenario based on an instance of this problem in section 4.1. In the rest of this chapter, we overview a generic model for the Online Localized Resource Allocation (OLRA) [167] in section 4.2. Then, we propose in section 4.3 an extension to this model dedicated to our problem with the specification of communication and additional time constraints.

4.1 Illustrative Scenario

Let us assume the simple scenario illustrated in Figure 1.2 on Page 10. We have a fleet V of connected autonomous vehicles distributed through the city: each vehicle $v \in V$ is defined by its capacity, cost per distance unit and average speed. The city road network is represented by a complete graph. At any point in time, vehicles can communicate by messages, and share information about their current situation:

- **location:** defines where the vehicle is currently situated in the road network graph,
- **load status:** specify the current number of free seats,
- **planning:** the vehicle knows its schedule for a specific period of time (next 24 hours for example), and may share partial or full information about it.

We consider that vehicles can communicate via an ad-hoc Vehicle-to-Vehicle (V2V) network, in which the communication range of vehicles could be limited. Thus it is important in such cases to consider communication robustness, which require efficient scalable network management.

Passenger requests occur online in a non-predictable manner. Each request takes the form of a query to travel defining

- **pickUp:** the origin location of the request from which the passenger is picked,
- **dropOff** : the passenger destination,
- **datetime** : the desired service time,
- **type** : defines if the request is specified by desired delivery time (DDT) or by desired pick-up time (DPT),
- **time window:** defines the maximum accepted deviation on the upper bounds on the actual (scheduled) time of the proposed solution from the desired `datetime`. i.e. it defines the maximum accepted delay of the service from the desired time.

Requests may also specify a set of constraints such as: **trip duration** that defines the maximum in-vehicle time, and **budget** that defines the maximum accepted trip cost.

A solution to this problem is defined by a set of proper schedules for vehicles that allow to serve a set of requests satisfying their constraints. An optimization of this solution may target maximizing the number of satisfied requests, minimizing waiting time (QoS objective), maximize profit by minimizing vehicle travel costs (QoB objective) or a combination of these objective that may increase the level of satisfaction of the users.

This problem can be expressed as a MARA problem in which we have a set of *Consumers* (vehicles) aiming at *consuming* (serving) a set of *Resources* (requests) under a set of constraints in dynamic settings.

4.2 OLRA Problem Model

In the context of transportation problems, Zargayouna et al. [167] proposed the Online Localized Resource Allocation (OLRA) problem considering both the geographical location of consumers and resources and their online appearance concurrently. Resources and consumers appear in a non-deterministic manner and can, subsequently, change their position at any time. Resources can be, for instance, vehicle seats, parking lots, electric car charging places, etc. Each resource is defined by a set of properties and has a dynamic state. An allocation changes the resource state, but the resource-defining properties remain the same. A consumer typically starts searching for a resource at non-deterministic time instances, i.e. the appearance time and availability period of resources are neither predefined nor predictable for consumers but discovered during the process.

The compliance of a resource with the requirements of the consumer depends on their spatial and temporal situations, as well as on the consumer constraints and preferences for the state and properties of the resource (as in all resource allocation problems). A consumer preference for resources is measured with an individual utility function. The local objective of consumers is to maximize their own utility, while the global objective is generally to minimize the total distance traveled and/or the total travel time.

4.2.1 Problem Formulation

An OLRA problem is defined as a tuple:

$$OLRA = \langle \mathcal{R}, \mathcal{C}, \mathcal{G}, \mathcal{D} \rangle \tag{4.1}$$

where: $\mathcal{R}$ is the finite set of resources, $\mathcal{C}$ is the finite set of consumers, $\mathcal{G} = \langle \mathcal{N}, \mathcal{E} \rangle$ is a graph, with a set of nodes $\mathcal{N}$, a set of edges $\mathcal{E} = \{e_{ij} | i, j \in \mathcal{N}, i \neq j\}$, and $\mathcal{D} = \{d_{ij} \in \mathbb{R}+ | e_{ij} \in \mathcal{E}\}$ is the set of distance values d_{ij} associated with each edge $e_{ij} \in \mathcal{E}$.

The distance between two adjacent nodes is fixed, while the travel times may vary according to the environment dynamics. At any time, one or more resources from $\mathcal{R}$ can be located on the same node.

Environment Dynamics

The following two functions define the dynamic travel time between nodes of $\mathcal{G}$ and the dynamic positions of resources and consumers.

$$\tau : \mathcal{E} \times \mathcal{T} \rightarrow \mathbb{R}+ \tag{4.2}$$

$$\rho : \mathcal{R} \cup \mathcal{C} \times \mathcal{T} \rightarrow \mathcal{N} \tag{4.3}$$

Where $\mathcal{T}$ is the time domain (time horizon), and $\tau(e, t)$ is the trip duration to cross an edge e at time $t \in \mathcal{T}$. The trip duration to go from node i to node j ($e_{ij} \notin \mathcal{E}$) is the sum of duration of the edges on the shortest path at t. $\rho(r, t)$ (resp. $\rho(c, t)$) is the node where a resource r (resp. a consumer c) is located at time t. A resource r or consumer c moving on edge e_{ij} is considered positioned on i until it reaches j. Thus, to determine whether a resource $r \in \mathcal{R}$ and a consumer $c \in \mathcal{C}$ are located at the same node or not, the following function is defined:

$$\texttt{same\_location}: \quad \mathcal{R} \times \mathcal{C} \times \mathcal{T} \rightarrow \{0, 1\} \tag{4.4}$$

$$\texttt{same\_location}(c, r, t) = \begin{cases} 1 & \texttt{if } \rho(c, t) = \rho(r, t), \\ 0 & \texttt{otherwise} \end{cases}$$

At time t, for a consumer c and a resource r, `same_location` is a boolean function whose value at time t is 1 if and only if the consumer location $\rho(c, t)$ is the same as the resource location $\rho(r, t)$.

Availability

The availability of a consumer or a resource is determined by its dynamic status attributes; a parking spot which is occupied by a vehicle at some time instance t is an example of unavailable resource, similarly, a vehicle (as a consumer) being in pan or having maximum number of passenger on-board is considered unavailable to consume more requests. To represent the non-deterministic availability of a consumer or resource, the following function is defined:

$$\texttt{available} : (\mathcal{R} \cup \mathcal{C}) \times \mathcal{T} \rightarrow \{0, 1\} \tag{4.5}$$

If the resource r (or consumer c) is available at time t for the allocation process, the value of $\texttt{available}(r, t) = 1$ (resp. $\texttt{available}(c, t) = 1$).

Compliance

The function $\texttt{compliant}(c, r)$ defines whether or not the values of the properties of the resource r correspond to the requirement of the consumer c, in which case its value is 1. A value of 0 means that the resource r cannot be assigned to consumer c.

$$\texttt{compliant} : \mathcal{C} \times \mathcal{R} \times \mathcal{T} \rightarrow \{0, 1\} \tag{4.6}$$

4.2.2 Allocation Constraints and Objectives

A solution to an *OLRA* instance is an allocation of resources to consumers. The determination of a solution is based on the function γ, which specifies whether a consumer actually consumes a resource at a particular time:

$$\gamma : \mathcal{C} \times \mathcal{R} \times \mathcal{T} \rightarrow \{0, 1\} \tag{4.7}$$

$\gamma(c, r, t)$ takes value 1 if a consumer c uses a resource r at t, 0 otherwise.
Allocating a resource r to consumer c at t is constrained by the status of both c and r at t (i.e. their positions, availability and compliance) the allocation feasibility is defined by the following function:

$$\texttt{feasible} : \mathcal{C} \times \mathcal{R} \times \mathcal{T} \rightarrow \{0, 1\} \tag{4.8}$$

where:

$$\begin{aligned} \texttt{feasible}(c, r, t) = \quad & \texttt{same\_location}(c, r, t) \times \texttt{available}(c, t) \\ & \times \ \texttt{available}(r, t) \times \texttt{compliant}(c, r, t) \end{aligned} \tag{4.9}$$

A consumer c cannot consume a resource r at t unless $feasible(c, r, t) = 1$.

$$\gamma(c, r, t) \leq \texttt{feasible}(c, r, t) \quad \forall c \in \mathcal{C}, r \in \mathcal{R}, t \in \mathcal{T} \tag{4.10}$$

Besides, this allocation is also constrained by the resource shareability and how many resources a consumer can take simultaneously.

$$\sum_{c \in \mathcal{C}} \gamma(c, r, t) \leq k_r, \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{4.11}$$

$$\sum_{r \in \mathcal{R}} \gamma(c, r, t) \leq l_c, \forall c \in \mathcal{C}, \forall t \in \mathcal{T} \tag{4.12}$$

Constraints[1] (4.11) specify that the sharable resource r can be simultaneously taken by at most ($k_r \in \mathbb{N}$) consumers; the case $k_r = 1$ means the resource is not shareable. Constraints (4.12) specify the consumption capacity of the consumer c in terms of the maximum number of resources ($l_c \in \mathbb{N}$) that it can take simultaneously.

Utility and Local Objectives

The aim of every individual consumer is to consume the resources that maximize its benefit. The benefit of allocating a resource to a consumer is given by the following utility function:

$$\mu : \mathcal{C} \times \mathcal{R} \times \mathcal{T} \rightarrow \mathbb{R}+ \tag{4.13}$$

$\mu(c, r, t)$ is the utility value of allocating a resource r to consumer c at time t. As a consequence, the local objective of a consumer c is to maximize the sum of the utilities $\mu(c, r, t)$ associated with being allocated some resources r all along the time horizon ($\forall t \in \mathcal{T}$), i.e. of the utilities associated with allocation choices $\gamma(c, r, t) = 1$.

Global Objective

The global objective of the solution for an *OLRA* problem instance is generally to minimize the total operational costs which can be expressed in terms of the consumers' travel distance

[1] the relation 4.11 defines a set of constraints, each is applied for a couple (r, t) and so relation 4.12 for (c, t)

and travel time. The following functions, δ and φ represent the total distance traveled by the consumer, and its total travel time.

$$\delta : \mathcal{C} \to \mathbb{R}+ \tag{4.14}$$

$$\varphi : \mathcal{C} \to \mathbb{R}+ \tag{4.15}$$

The operational cost of a consumer c is calculated as the weighted sum

$$\texttt{cost(c)} = \alpha\overline{\delta(c)} + \beta\overline{\varphi(c)}$$

where $\overline{\delta(c)}$ and $\overline{\varphi(c)}$ are normalization of $\delta(c)$ and $\varphi(c)$ to express the function terms in the same unit, α and β are positive numbers weighting the relative importance of time and space in the specific problem that is considered. α and β are considered as integrate scaling factors to reflect the weights of time and space s.t. $\alpha + \beta = 1$, i.e. they encode the cost per distance and cost per time as probability to keep normalized scale for `cost(c)`.
Thus, the global objective function is expressed as minimizing the sum of the operational costs for all consumers:

$$\min \sum_{c \in \mathcal{C}} \texttt{cost(c)} \tag{4.16}$$

However, A system may perform well on the consumer's local objectives while the global is not optimized, or it may show good results for the global objective while the individual goals are of poor quality. This could happen because the allocation decisions for a consumer affect the status of resources and thus affect the feasibility of other consumers allocation options. Therefore, a significant increase in a consumer benefit could result in a decrease in the benefit of others, and possibly an increase in the total operational cost. As usual in this type of problem, there is a trade-off between these goals that should be found via proposed solutions.

4.3 Extension: AV-OLRA Model

To map the *OLRA* problem model to the illustrative scenario we expounded in section 4.1, we propose the *AV-OLRA* problem model, a specialization of *OLRA* with autonomous vehicles. *AV-OLRA* extends *OLRA* with the communication and additional time constraints modeling.

Defining a time sampling unit `tick` we formulate the AV-OLRA problem as follows:

$$\text{AV-OLRA} := \big(\mathcal{R}, \mathcal{V}, \mathcal{G}, \mathcal{T}\big) \tag{4.17}$$

where the set of resources $\mathcal{R}$ defines a dynamic set of trip requests that occur to be available for a specific time window during the execution; the set of consumers $\mathcal{V}$ represents a fleet of m autonomous vehicles that are mobile and can only communicate within a limited range ($\mathcal{V}$ specialize $\mathcal{C}$ in *OLRA*); $\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \omega \rangle$ is a graph, with $\mathcal{N}$ the set of nodes, $\mathcal{E}$ the set of edges, and ω is a valuation function that associates each edge $e \in \mathcal{E}$ with the value based on a temporal distance measure (e.g., average driving time in `ticks`), which will be used to calculate the operational costs of vehicle trips. Here the graph $\mathcal{G}$ represents an urban road network in which ω encodes $\mathcal{D}$ from the original *OLRA* in terms of trip duration instead of spatial distance. We enforce the time horizon of the problem $\mathcal{T}$ to be expressed as a discrete list of `ticks` $\mathcal{T} = \{t_0, t_1, \ldots, t_{end}\}$ which simplifies the computation of schedules and time conflict detection.

Definition 1 *A trip request $r \in \mathcal{R}$ is characterized by four constant properties:*

- *origin $o_r \in \mathcal{N}$ defining the pick-up location*
- *destination $d_r \in \mathcal{N}$ defining the drop-off location*

- *a time-window* $tw_r[l_r, u_r]$ *defining the lower- and upper-bound of accepted service time*
- *size* $s_r \in \mathbb{N}^+$ *specifying the number of required seats*

and has a dynamic (time-dependent) property denoting for every time tick t its status $st_r^t \in \{\texttt{announced, available, picked, satisfied, delayed, expired}\}$

$$r := \left(o_r, d_r, tw_r, \texttt{size}_r, st_r^t\right) \tag{4.18}$$

Definition 2 *An autonomous vehicle* $v \in \mathcal{V}$ *is characterized by three constant properties*

- *capacity* $\texttt{c}_v \in \mathbb{N}$
- *driving cost per traveled distance* $\texttt{cpd}_v \in \mathbb{R}+$
- *a limited communication range* $\texttt{rng}_v \in \mathbb{R}+$

It has also a set of time-dependent properties which are

- *current location* $\texttt{loc}_v^t$
- *current destination* $\texttt{dest}_v^t$
- *the number of currently available* $\texttt{seats}_v^t$
- *last taken decision* $\texttt{dec}_v^t$
- *the* $\texttt{request}_v^t$ *on-board if exists*

$$v := \left(\texttt{c}_v, \texttt{cpd}_v, \texttt{rng}_v, \texttt{loc}_v^t, \texttt{dest}_v^t, \texttt{seats}_v^t, \texttt{dec}_v^t, \texttt{request}_v^t\right) \tag{4.19}$$

4.3.1 Connected Sets and Sub-problem Instances

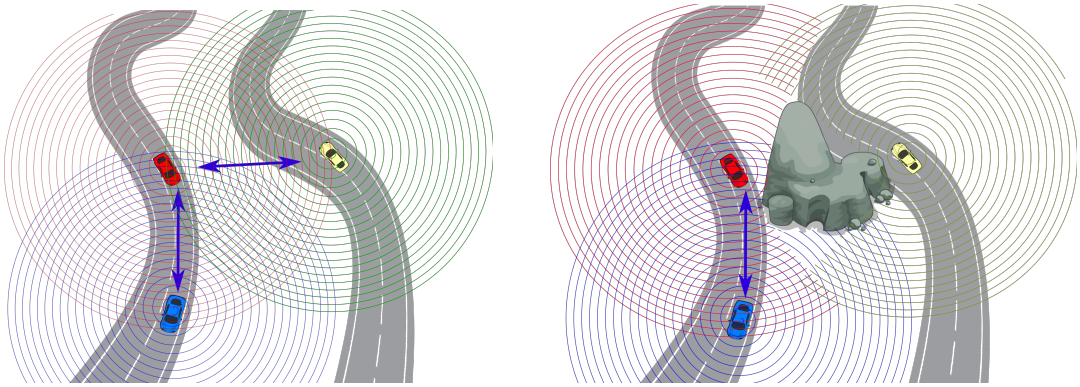
Vehicle connectivity is mainly constrained by the communication range. The intensity of radio waves over distance obeys the inverse-square law of electromagnetic propagation, which accounts for loss of signal strength over distance.
For each vehicle $v \in \mathcal{V}$ the communication range $\texttt{rng}_v$ defines the maximum distance to which v can send messages (shown in Figure 4.1a). The communication range value depends only on the used communication technology standard. Considering it in our model adds another dimension of genericity. However, radio signal drops not only because of distance but also because of obstacles like buildings, mountains, and tunnels (as shown in Figure 4.1b).

The binary function `d_ctd` defines if two vehicles are connected directly to each other.

$$\texttt{d\_ctd} : \mathcal{V} \times \mathcal{V} \times \mathcal{T} \rightarrow \{0, 1\} \tag{4.20}$$

Assuming the absence of obstacles and other environmental factors that may lead to blocking or distorting the signal, the connectivity between two vehicles $v \in \mathcal{V}$ is achieved if the distance between them is less than or equal to the minimum of their communication ranges. Considering a function $\texttt{distance} : \mathcal{N} \times \mathcal{N} \rightarrow \mathbb{R}^+$ that returns the euclidean distance between two points $(i, j \in \mathcal{N})$, we can express `d_ctd` as follows:

$$\texttt{d\_ctd}(v_1, v_2, t) = \begin{cases} 1, & \text{if } \texttt{distance}\left(\texttt{loc}_{v_1}^t, \texttt{loc}_{v_2}^t\right) \leq min(rng_{v_1}, rng_{v2}) \\ 0, & \text{otherwise} \end{cases}$$



(a) Ideal settings: No obstacles (b) obstacle reduces connectivity

Figure 4.1: The limited range connectivity

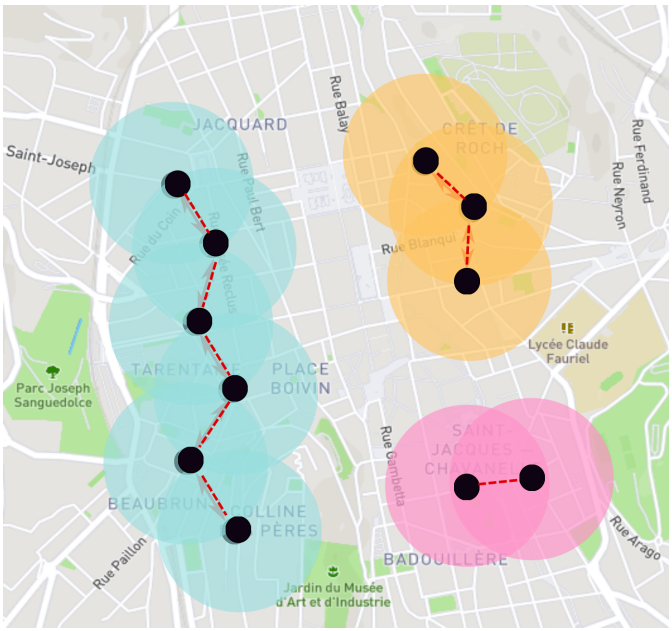


Figure 4.2: The transitive connectivity and connected sets

However, two vehicles v_1 and v_2 being too distant from each other w.r.t. their communication ranges, can still be aware of each other via other intermediate vehicles and connected transitively. This is achieved if there exists v' that is connected directly or transitively to both of them. The binary function `ctd` generalizes the `d_ctd` with the transitive connectivity.

$$\texttt{ctd} : \mathcal{V} \times \mathcal{V} \times \mathcal{T} \to \{0, 1\} \tag{4.21}$$

$$\texttt{ctd}(v_1, v_2, t) = \begin{cases} 1, & \text{if } \texttt{d\_ctd}(v_1, v_2, t) = 1 \\ & \text{or } \exists v' \in \mathcal{V} \setminus \{v_1, v_2\} : \texttt{ctd}(v', v_1, t) \times \texttt{ctd}(v', v_2, t) = 1 \\ 0, & \text{otherwise} \end{cases}$$

This leads to the definition of connected sets:

Definition 3 *The Connected Set (CS) of a vehicle is the set of vehicles that are connected to it directly or by transitivity.*

$$CS : \mathcal{V} \times \mathcal{T} \to 2^{\mathcal{V}} \tag{4.22}$$

$$CS(v,t) = \{v' \in \mathcal{V} \mid \texttt{ctd}(v,v',t) = 1\}$$

Connected sets are dynamic entities; they are created, split, merged at run-time based on the movement of the vehicles. Thus, based on the previous definitions, a vehicle may communicate at time t only with the members of its connected set. The limited communication ranges implicitly partitions the fleet into multiple connected sets (see Figure 4.2). The shorter the communication ranges are, the smaller, the more connected sets exist. A fleet with long enough communication range could result in only one connected set with global knowledge sharing. Decentralized decision-making is an operation that is highly correlated to natural conflicts [82]. When resource allocation decisions are made, an important challenge is that these decisions can sometimes conflict with each other; for example, when two consumer agents simultaneously plan to use the same resource. In such circumstances, consumers must manage these conflicts. Conflict management approaches in MAS can be aimed at either avoiding conflicting decision or detecting and resolving the existing conflicts. The key to conflict avoidance is to have a method to identify potential conflicts before making a decision, while conflict resolution approaches allow consumers to first make decisions and then use a method to detect and resolve the resulting conflicts when exist.

Definition 4 *A solution for AV-OLRA for a connected set is an aggregation of the allocations of requests to all vehicles in this set, in which each request is allocated to at most one vehicle (i.e. absence of allocation conflicts).*

Definition 4 implies that a solution to an AV-OLRA instance defined for the vehicles and the requests may be sub-optimal because several vehicles consider the same request or because the optimal solution is not the union of the optimal solutions in each connected set. Also, any solution is time-dependent according to the online dimension of the problem.

4.3.2 Quality of Resource Allocation

The quality of an allocation is characterized by functional and technical indicators whose computation is independent of solution approaches, and can therefore help compare suitability of these approaches. The functional indicators are measures of optimality of the allocation process defined by its objective function, while the technical indicators are used to assess the feasibility and applicability of the allocation process and to predict its costs in different settings. In this work, we characterize the quality of AV-OLRA solutions in ODT scenarios by the following indicators:

Quality is the *percentage of satisfied (consumed) requests* from all announced requests known by the agents. Therefore, this indicator points to the Quality of Service (QoS) level.

Utility is the *total utility* of schedules from a global point of view. It points to the total revenue of the fleet, the calculation of which is derived from the distances of successful trips (driven with passengers on board from source to destination) in addition to the fixed service fee per served request, which defines the profit for the ODT service provider.

Cost is the *operational cost*, derived from the total driven distances of the vehicles.

MsgCount is the *total number of messages* exchanged during the allocation process.

MsgSize is the *average size of messages* exchanged during the allocation.

The relation between **Utility** and **Cost** indicators defines the Quality of Business (QoB). The two last communication indicators can be used to estimate the technical cost of the solution and predict if such a solution is applicable in terms of communication, i.e., if it could cause critical communication overload.

4.3.3 Utility, Constraints and Objectives

In this model, we define the utility function of a vehicle based on the indicators of the quality of solution described in section 4.3.2. The vehicles aim to satisfy as many requests as possible in order to maximize their utility. Hence, if we consider a local version of the **Quality** indicator calculated individually for each vehicle $v \in \mathcal{V}$ we can define the vehicle utility in terms of the percentage of request allocated to v among its all known requests. Allocating a request r to a vehicle v is constrained by the spatial and temporal availability of both v and r. We consider the origin and destination of requests to be constants, and a request is available to pick-up only at its origin during its defined time-window $tw_r = [\mathtt{l}_r, \mathtt{u}_r]$. Thus, allocating r to v requires that v can arrive in origin point of r at a time t in between the lower-bound $\mathtt{l}_r$ and the upper-bound $\mathtt{u}_r$ that define the time-window of r.

Whether car-sharing scenarios are considered or not, the fleet of vehicles could be heterogeneous, thus the request size (required seats) and the vehicle capacity should be taken into account. This implies that the compliance function is subject to the capacity constraint:

$$\exists t \in tw_r : \quad \mathtt{seats}_v^t \geq \mathtt{size}_r \tag{4.23}$$

Thus the feasibility of allocating a request r to a vehicle v is then subject to finding a time t at which they both are available and existing at the same location, and to have enough number of seats in v to pick r.

From a global perspective of ODT as a business model, the main objective of ODT service providers is optimize their benefits by reducing costs and raise the profit. From this point of view we can define the objective function $\mathcal{F}$ to be maximized by the allocation process based on the relation between the **Utility** and **Cost** indicators:

$$\mathcal{F} = \sum_{r \in \mathcal{R}_s} (P + p \cdot \mathtt{dist}(r)) - \sum_{v \in \mathcal{V}} \mathtt{cpd}_v \cdot \mathtt{driven}(v) \tag{4.24}$$

where $\mathcal{R}_s \subseteq \mathcal{R}$ is the set of all satisfied requests, P is a fixed price (service fee) per request, p is a pricing factor per unit of travelled distance, $\mathtt{dist}(r)$ is the total trip distance for a request r and $\mathtt{driven}(v)$ is the total driven distance by v.

Summary

This chapter aimed at proposing a model for a resource allocation problem encountered in the management of autonomous vehicle fleets. We define the AV-OLRA problem model, a specialization of OLRA with autonomous vehicles, and an extension with the modeling of additional communication and time constraints.

Taking into consideration the limited communication range of vehicular ad-hoc networks, we defined the concept of connected sets, in which vehicles can exchange direct and transitive messages and thus coordinate between each other about their allocation decisions.

Our model is well-suited to the ODT domain, where fleets respond to passenger requests in dynamic online environments, and can handle different types of constraints and allow different approaches to find solutions to coordinate vehicles.

Chapter 5

Multiagent Approach to AV-OLRA

This chapter presents the MAS architecture to deploy the Cyber-Physical System (CPS) of AV-OLRA. Recalling its definition in Chapter 2, a Multiagent system consists of Agents and their Environment. Agents are autonomous entities that decide and act in the system by interacting together and with the resources in the shared environment. The environment is the place in which the agents are situated and can move following its topology. It is also the place where resources and tools (passive or active entities that we call artifacts from now on) are located. In what follows, we describe the main components that are essential to model and program a MAS for AV-OLRA. These interacting components that share information through Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I) and Pedestrian to Infrastructure (P2I) communication modes are illustrated in Figure 5.1.

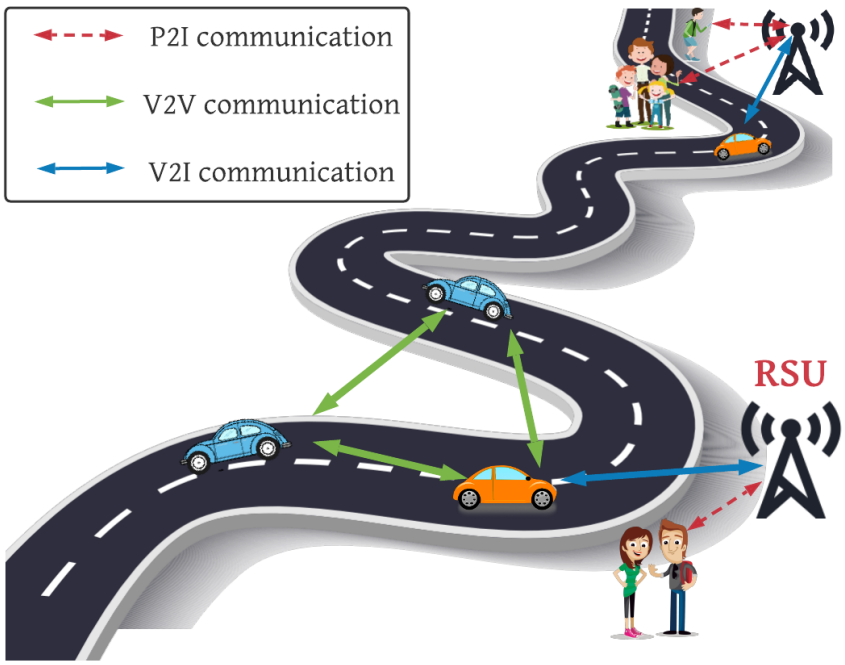


Figure 5.1: AV-OLRA multiagent system components and their interaction

5.1 Environment

The environment is defined on two topologies:

- **Physical:** represent the spatial urban network represented by the graph $\mathcal{G}$ of *AV-OLRA*.
- **Communicational:** represent the message exchange and routing infrastructure, in terms of a scalable dynamic connection graph that is composed of all connected sets (*CSs*)

Unlike agents, artifacts in the MAS environment are considered as non-autonomous decision entities. Agents act on artifacts through artifact-owned operations and can perceive some of their states and properties. An artifact is described by a set of properties that could be perceived by agents and by a set of operations that are accessible to agents to act on the environment. The types of artifacts that are found in this dynamic environment are described in the following sections.

5.1.1 Source Artifact

The request emission sources are non-mobile road-side units (RSUs) distributed over the urban spatial network. The source artifact is responsible for emitting trip requests that could be collected by vehicles. When emitted, the request information remains active (i.e., continuously announced in the source communication range) until it is picked up or expires when the upper-bound of its time-window is exceeded. The source artifact is not interactive (i.e., it does not provide any operations to the agents). However, it provides agents with one property:

- *requests:* a list that contains information about active requests, including their constant and time-dependent properties (see Definition 1 on page 39).

5.1.2 Vehicle Artifact

As a physical object a vehicle has non-autonomous parts that can be seen as an artifact composed of the on-board units to provide the driving control and communication functionalities to the autonomous vehicle agent (see Figure 5.2). An agent can act on the Vehicle artifact to move in the physical environment from one location to another, stop, and communicate with agents in its connected set i.e. its communicational environment.

- **Vehicle properties:**

  The vehicle artifact has the following set of properties:

  - *location*: defines the location of the vehicle at a specific moment of time ($\texttt{loc}_v^t$)
  - *destination*: defines the vehicle's target location at a specific moment ($\texttt{dest}_v^t$)
  - *schedule*: defines the vehicle's plan as a list of steps (each step is a tuple representing a location to visit and its potential visit time).
  - *state*: defines the operational state of vehicle at a specific point of time ($\texttt{state} \in \{\texttt{moving}, \texttt{waiting}, \texttt{picking\_up}, \texttt{delivering}, \texttt{unavailable}\}$).
  - *request*: defines request of the passenger that is currently on-board ($\texttt{request}_v^t$), if any.
  - *seats*: defines vehicle capacity by the number of free seats at a specific point of time ($\texttt{seats}_v^t$)

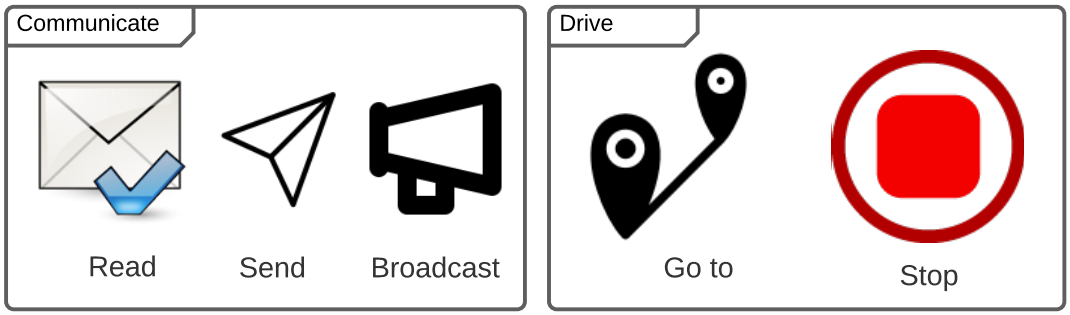


Figure 5.2: Vehicle on-board unit interface

- *connected*: defines a list of entities with whom the vehicle can communicate (i.e. the members of its connected set) at a specific point of time (see equation 4.22)
- *known*: defines the set of requests that the vehicle is aware of at a specific moment of time, it can be expressed as $\texttt{known}^t_{v_i}$.

One can notice that some of these properties correspond to the time-dependent properties from the vehicle definition (Definition 2) while others are specific to the implementation of the CPS.

- **Vehicle operations:**

The vehicle artifact provide the following set of operations that its agent can invoke:

- `go_to`(l): activating this operation on a vehicle v at time t changes the property $\texttt{dest}^t_v$ to l, sets the vehicle *state* to `moving` and forces the vehicle to start moving in the spatial network toward the new target, obviously this will change dynamically the value of $\texttt{loc}^t_v$.
- `stop`(): terminates vehicle movement in the spatial network, changes the value of $\texttt{dest}^t_v$ to `NULL` and changes the vehicle *state* to ***`waiting`***.
- `send(receiver, topic, payload)`: sends through the communication network a message composed of a topic, and a payload (i.e. message body) to a receiver belonging to the connected set of the vehicle.
- `broadcast(topic, payload)`: sends through the communication network a message composed of a topic and payload to all agents belonging to the connected set of the vehicle.

5.1.3 Resource Artifact

This artifact has all properties which are common to any type of consumable resources situated in physical environment. Resource artifacts may represent any object that has constrained and limited availability to be consumed by an agent. For instance, we may have artifacts for parking slots, tolls, transport requests, charging/fuel stations, or any other facility. In this work, we are mainly interested in trip requests as resources. A trip request resource artifact is a virtual entity representing the properties of the passengers (or transported objects), the request dynamic status and its interaction model (Equation 4.18).

- **Request resource properties:**
  - *resourceInfo*: a description in the form of a list of tuples (key, value) representing static properties that characterize the resource; in the case of trip request resource,

its origin o_r, destination d_r, size $\texttt{size}_r$, time_window tw_r, and other optional static constraints.

- *status*: the dynamic status of the resource; in the case of a trip request this property refers to the st_r^t.
- *location*: the position of the resource in the urban network at a given time (Equation 4.3).

- **Request resource operations:**
  The operations of resource artifacts depend on the resource type and define the functionalities that enable agents to interact with, benefit from and consume the resource. For trip request resources we define the following operations.
  - `pick_up`(): performing this operation on a request r at a time t requires that its *status* st_r^t = `available`. Once activated, this will changes st_r^t to `picked`. After being `picked`, the resource artifact moves along with the vehicle, which means that its *location* property will change dynamically.
  - `drop_off`(): the *status* of the request artifact r must be `picked` at the time t when this operation is performed (st_r^t = `picked`). As consequences, if the time constraints of the request is violated, the *status* st_r^t is set to `delayed`, otherwise st_r^t is set to `satisfied`.

5.2 Agents

In this work we consider only one type of agents. An autonomous vehicle (AV) agent is associated with each vehicle in the system. We can distinguish three different sub-behaviors (*acting*, *communicating*, and *planning*) of an AV. As we model AV-OLRA in discrete time space, the time horizon is defined as a set of ticks. At each time tick every agent performs the following actions as shown in Figure 5.3 where the abstract elements are to be implemented according to the specifications of the solution method:

1. read the received messages and update the context (communicating sub-behavior)
2. choose the locations to visit (planning sub-behavior)
3. act by performing a driving action (acting sub-behavior)
4. broadcast its context information (communicating sub-behavior)

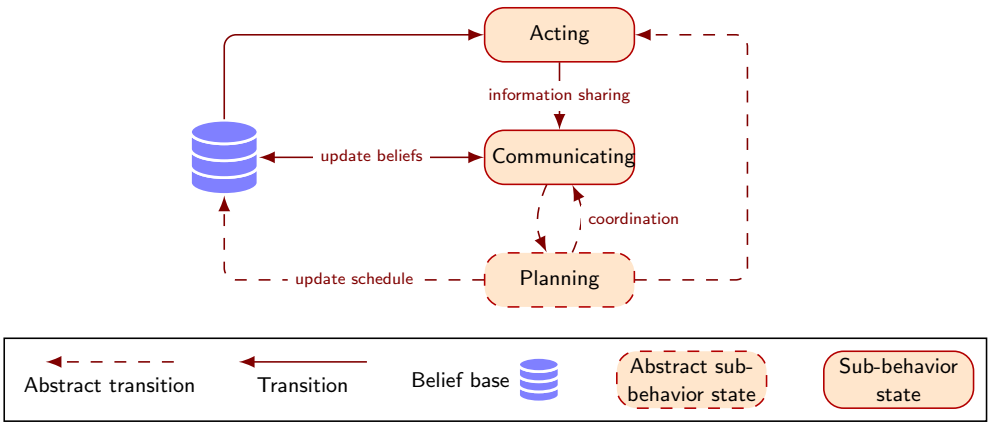


Figure 5.3: Generic vehicle behavior

5.2.1 AV Acting Sub-behavior

Based on the presence of passengers on-board, the vehicle location, and its knowledge about upcoming requests, in the acting sub-behavior the AV agent can be in one of the following states, as shown in Figure 5.4:

Marauding the vehicle has no passenger on-board and is looking around for its next destination;

Moving the vehicle has a destination and is moving in the urban topology towards this destination;

Picking up the vehicle is located at the origin location of the passenger p's request to perform the `pick_up`(p) action and then start moving again;

Dropping off the vehicle is located at the destination location of the passenger p's request to perform the `drop_off`(p) action and then look for a new destination.

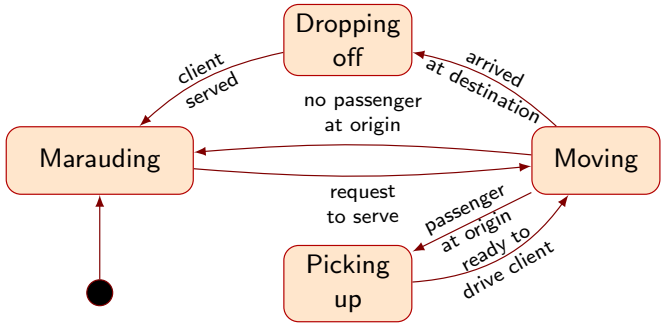


Figure 5.4: Acting sub-behavior

The *AV* acting states represent the operational states of the *vehicle artifact* except for *unavailable* in which the *acting* sub-behavior is never triggered. the *Marauding* is a state defining that the vehicle has no information about available request, thus it is waiting to receive new messages to decide its next destination. This can be realized either in communication topoloy only or in both topologies. In the first case the state of the vehicle artifact is *waiting* until receiving new request messages either from a nearby source or from another vehicle that joins its connected set. In the later case, the vehicle artifact is *moving* in the spatial network towards random neighboring destinations to increase the probability of being in the range of other vehicles and thus the probability of receiving new information messages. In the specific context of this work, we consider the second case.
Transitions between acting sub-behavior states are based on the following set of events (shown in Figure 5.4):

***request to serve*:** the vehicle v has no passenger on board and a request r is selected to be served, this implies the vehicle has a new destination to go to;

***passenger at origin*:** the vehicle v arrived at the origin location of request r, o_r, and the passengers of r are present in o_r so v can start to pick up;

***ready to drive client*:** the passengers of a request r got on vehicle v that is now ready to drive to the destination location d_r;

***arrived at destination*:** the vehicle v has passengers on board of a request r and arrived at the destination location d_r;

***client served*:** the vehicle v is free to choose a request to serve, after delivering passengers at their destination, and may look around for the next request to serve;

***no passenger at origin*:** the vehicle v has no request on board, arrived at the origin o_r of a request r, but the passengers of r are absent. the vehicle becomes then free again.

5.2.2 AV Communicating Sub-Behavior

As communicating agents, AVs have a communication behavior with other surrounding entities through the dynamic connection scheme (Connected sets) illustrated in Figure 4.2 ; they can join/leave connected sets, broadcast, send, and receive messages:

- `join`(c): an agent joins a connected set c as a result of being in the communication range of one of its members;
- `leave`(c): an agent recognizes that it is leaving its connected set c as a result of being disconnected from all its members;
- `send`(m, a): an agent sends a message m to another agent a, provided that they are in the same connected set;
- `receive`(m): an agent receives a message m from another agent in its connected set (once received and read, the message is stored in the agent belief base);
- `broadcast`(m) similar to `send`(m, a) but here the agent doesn't specify the receiving agent, instead it broadcasts the message to all the other members of the connected set.

A message has a source (the sender identifier) a target (the recipient identifier in case of direct message or "broadcast" otherwise), a topic (a character string defining the subject), and a payload (the message body content). Table 5.1 lists a set of communication event examples and their corresponding message specifications. The message payload is processed by the target agent based on its topic. For instance, on receiving a message with "*new_req*" topic, the agent v add the information of the request r from the message payload to its set of known request R_v. the same happens for the list of requests received in the payload of messages with "*known*" and "*join*" topics.

Event	message			
	source	target	topic	payload
new request r	source artifact	broadcast	*new_req*	$\left(o_r, d_r, \texttt{type}_r, tw_r, \texttt{size}_r, st_r^t\right)$
v' joins a CS	v'	broadcast	*join*	$\mathcal{R}_{v'} = \{r \in \mathcal{R} : v' \texttt{ knows } r\}$
`received:` (v',*join*,$\mathcal{R}_{v'}$)	$v \in CS$	v'	*known*	$\{r \in \mathcal{R} \setminus \mathcal{R}_{v'} : v \texttt{ knows } r\}$

Table 5.1: Generic message types and their specifications

5.2.3 AV Planning Sub-Behavior

The planning sub-behavior of AVs depends on the chosen coordination mechanism that could be centralized/decentralized (see Section 2.1.4) in which agents can adopt cooperative or competitive behaviors (see Section 2.2.3). Figure 5.5 illustrates the common components of the generic planning sub-behavior of AVs. The implementation specifications of these components for instances of AV depend on the adopted coordination mechanism, they are detailed in Part III. Generally speaking, an agent has at each time tick a set of planning options (e.g. add a request to schedule, abandon a request, make an exchange offer to another vehicle, ..). The set of available options changes dynamically based on the current context of the problem, vehicle status and its adopted coordination mechanism. For updating its schedule, an AV continuously

looks for planning options. If any option is found, the AV selects one and depending on the coordination mechanism it communicates or not its decision to its neighbors belonging to the same CS. The CS members reach an agreement or disagreement, depending on the coordination mechanism and the selected option. On agreement, the AV updates its schedule and looks for the next option and so on until no option is available.

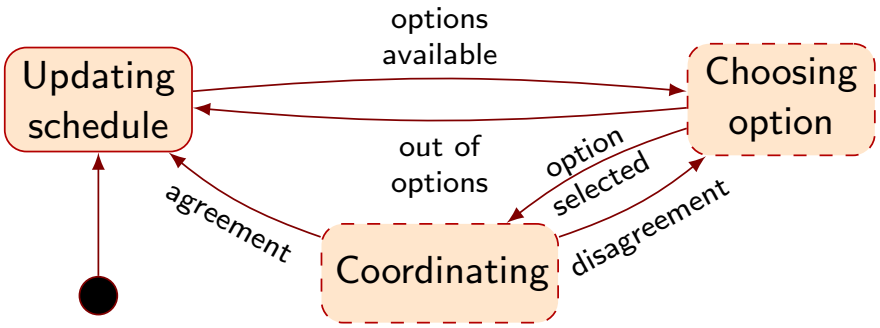


Figure 5.5: AV generic planning sub-behavior

5.2.4 AV Coordination Mechanisms

A coordination mechanism `CM` is defined by 3-tuple

$$CM = (DA, AC, AM) \tag{5.1}$$

where DA denotes the level of decision autonomy which is either centralized (C) or decentralized (D); AC denotes the agents cooperativeness level with (S) or without (N) sharing of schedule information, and AM is the allocation mechanism (see Section 2.1.4). Although the proposed model supports the application of several coordination mechanisms, in this document and for all experimental scenarios, we consider that agents of the same fleet are homogeneous, i.e. they have the same coordination mechanism to prevent any ambiguous action.

According to this formulation and based on the different types of solution methods presented in Chapters 2 and 3, we can instantiate our generic model to implement coordination mechanisms from literature like: classical *selfish* behavior $\langle D, N, \text{Greedy}\rangle$ van Lon et al. [155], centralized *dispatching* $\langle C, S, \text{MILP}\rangle$ El Falou et al. [53], Lee et al. [91], Yang et al. [162], *cooperative* teams using DCOP to coordinate $\langle D, S, \text{DCOP}\rangle$ Fioretto et al. [56], and *auction*-based allocation $\langle D, S, \text{Auction}\rangle$ Daoud et al. [41], Egan and Jakob [50].

Mechanism	Functional specifications
Selfish	Priority function heuristic Conflict detection and resolution protocol
Centralized dispatching	Selecting the dispatching responsible agent Context information gathering and solution broadcasting protocol
Cooperative	The decision coordination protocol Conflict detection, avoidance, and resolution
Auctions	Priority function heuristic Auctioneer selection and auction initiation mechanism Winner determination

Table 5.2: Functional specification for different coordination mechanisms

To implement any of these coordination mechanisms, it is necessary to specify the coordinated activities that could occur throughout the interaction between the agents. The coordination functional specifications are not generic but strongly dependent on the used strategy for solving AV-OLRA. We can thus instantiate our generic model to implement variety of coordination mechanisms, each of which defines its own functional specifications (See Table 5.2). In the following chapter we explain in detail these coordination mechanisms and their social rules interaction protocols, and functional specifications.

Summary

This chapter aimed at proposing a multiagent-oriented programming model and defining the requirements to deliver the AV-OLRA model in which agents can communicate with each other via radio channels using peer-to-peer messages.

The communication model supports direct, broadcast, and transitive message transmission and is based on the concept of connected sets.

We aimed to provide a generic model; the proposed MAS will offer genericity on both communication and coordination dimensions. On the one hand, the limited communication range defines an attribute for the problem that affects the level of connectivity and thus bounds the achievable centralization. On the other hand, being dependent on the allocation process, the choice of the planning sub-behavior of AVs defines the coordination mechanism that affects the dynamic spatial-temporal context of the problem instances.

Part III

Solution Methods (to AV-OLRA)

Chapter 6

Centralized Dispatching

Solving the AV-OLRA resource allocation problem consists in optimally allocating vehicles to requests. We mentioned in the previous chapter that such a problem can be solved by different methods and algorithms depending on the problem constraints, as well as the chosen coordination mechanism. Traditional solutions rely on *dispatching*, in which a special entity (so-called *dispatcher*) is responsible for assigning available vehicles to requests. Once computed, the optimal schedules are communicated to the vehicles. This chapter describes in details the *centralized dispatching* approach, detailing the communication architecture and the problem formulation as an *Integer Linear Program (ILP)*. Since the allocation process is centralized, in the coordination mechanism formulation (see equation 5.1), the decision autonomy of agent is centralized ($DA = C$). The role of the agent is to update its schedule based on what he receives from the dispatcher.

The rest of this chapter is organized as follows. In Section 6.1 we describe our proposed dispatcher-based coordination mechanism detailing the information gathering and dispatcher selection within the connected sets. In Section 6.2 we introduce a traditional formulation of the problem as an ILP. In Section 6.3 we present an extension to the previous formulation that takes into consideration solving a sub-problem per connected set whenever the problem context changes.

6.1 Agent Coordination and Connected Sets Architecture

In this centralized model, during the planning phase, the AVs continuously ask a communication entity to update their schedules. Therefore, the *coordinating* state of AVs consists of a request/response protocol after which AVs are sent new schedules as an agreement.

Theoretically, the centralization aspect of this decision problem makes it possible to address it as an *Integer Linear Problem* and more specifically as a DARP (see Chapter 1). Even the dynamics of the problem (online requests, variable fleet size, traffic problems, and other environment dynamics) have been dealt with in the literature [119]. However, these problems are NP-Hard: this means that in reality if a globally central (omniscient) dispatcher exists, the complexity of its task grows quickly with the problem size (size of the fleet and the number of requests), and so do its response times, making it very difficult to keep the pace of the ongoing execution.

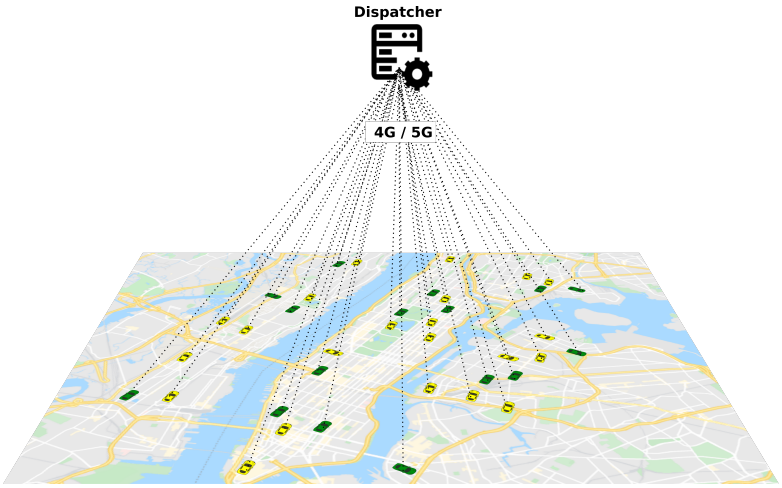


Figure 6.1: centralized dispatching via global communication

Moreover, querying the coordination entity according to the aforementioned request/response protocol requires vehicles to have continuous access to the dispatcher via a global communication infrastructure such as the Long-Term Evolution (LTE) communication or 5G (See Figure 6.1 in which the dispatcher is a communication portal), whose data traffic is expensive for the required usage density of such a persistent operation, and can cause a critical bottleneck on the dispatcher side. On the other hand, interacting through inexpensive means such as Dedicated Short-Range Communication (DSRC) implies constraining the vehicle communication by the hardware limitations of these means (e.g. the communication range). To handle these issues, we recall the Connected Set (CS) model for limited range communication presented in section 4.3.1 (at page 40).

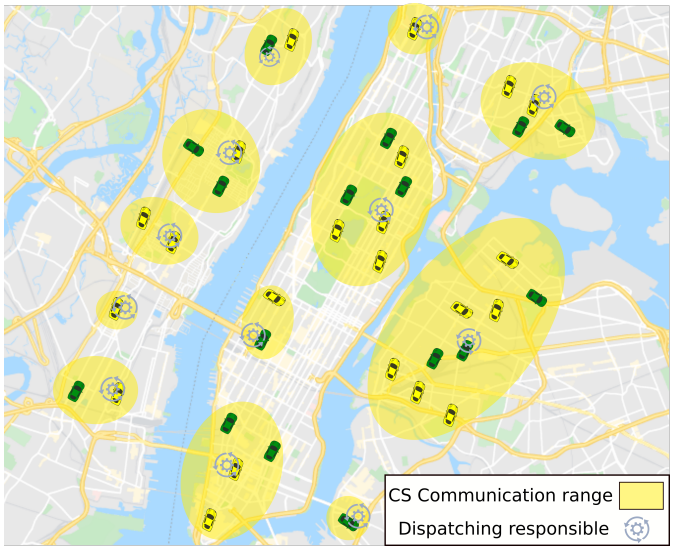


Figure 6.2: Dispatching per CS via limited range communication

We propose a model in which one dispatcher per CS is needed (as illustrated by Figure 6.2). Since AV-OLRA considers only one type of agents, the AVs, the dispatcher of a CS is one of its members: once a CS is created (or updated), one of its members becomes the dispatcher (see the following section). The chosen AV will be responsible for:

1. gathering the information from other agents about the requests ($AC = S$ according to the notation introduced in Section 5.2.4);
2. computing by ILP optimal schedules (or optimally updating previous ones) for the AVs of

the CS ($AM = MILP$);

3. sending to other vehicles of the CS their (currently) optimal schedules.

6.1.1 Selecting the Dispatcher Agent

Any agent in a *CS* can be selected as representative of all members to communicate with and play the role of the dispatcher (so-called *dispatcher agent*). A trade-off between communication bandwidth, choice stability, information robustness, and method responsiveness should be considered in the design of the method to choose a dispatcher agent.
Message exchange through transitive communication requires rerouting of messages through intermediary vehicles to their destination. The resulting data traffic may eventually lead to flooding or jamming in some areas of the network. The dispatcher agent should repeatedly receive context information from the *CS* members and send them their schedules. Hence, choosing one with maximum direct connectivity with other agents may effectively decrease the data traffic.

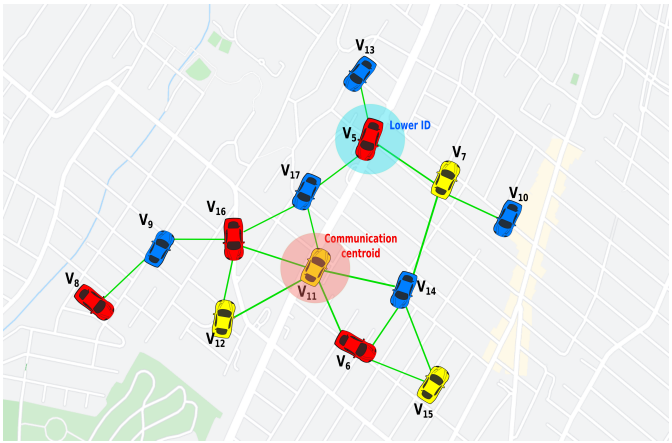


Figure 6.3: Communication centroid vs. lower ID dispatcher agent selection in a CS (Green lines represent the direct connection links)

If the dispatcher agent leaves the *CS*, a new one must be chosen among the remaining members. A new information-gathering round is then required in order to allow the new dispatcher to take over the dispatching task. According to Definition 3, *CS*s are dynamic entities that change, split, merge, and are created repeatedly based on the context dynamics. Therefore, the frequency and required computational time to select the dispatcher agent must be carefully taken into consideration.

Several strategies can be considered to choose the dispatching agent: here we will consider the *lower_id_agent* and the *centroid_agent* selection methods. In any selection method, if two or more agents in a CS share the highest value of the selection criteria, then the one having lowest id among them is selected to be the dispatcher randomly.

When they are first instantiated, agents are associated with ascending IDs. In the *lower_id_agent* strategy, the agent who has the lower ID is selected to be the dispatcher. One potential advantage of this method is the implementation and computation simplicity. On the other hand, the position of AVs in the communication topology of the *CS* is neglected. This could lead to a premature new dispatcher selection if the old one is located near to the *CS* border, and thus is more likely to leave it than others. Alternatively, one may consider the *centroid_agent* method to make a stabler choice. The centroid agent is positioned in the center of the *CS* topology, thus it has less probability to leave it and has more direct connectivity links with other members (as shows Figure 6.3).

A consequence of the *centroid_agent* selection method is that it requires more computational time, which will be added to the time to compute the new optimal routes immediately after the change in the CS status (which led to the dispatcher change). As the IDs of AVs are essential parts of exchanged message i.e. sources and targets (see Section 5.2.2), in this work and seeking implementation simplicity, we use the *lower_id_agent* method for our experiments.

6.1.2 Information Gathering

Due to the problem dynamics, agents must continuously share the context information, and questions about communication bandwidth and information robustness in the *CS* arise. Examples are: the method to choose dispatcher agent, or any situation in which an agent detects a change in his set of known request or in the urban network, and must report it by broadcasting a message to all members of *CS*.

To allow the dispatcher to calculate up-to-date solutions, all *CS* members must share with the dispatcher information about their current location, destination, available seats and current request on board if any. This information must be gathered by the dispatcher only, immediately prior to his computation of the solution (*pre-dispatching* messages). In addition to the generic message types presented in Table 5.1 on Page 50, some specific message types are required corresponding to information gathering and dispatching events. Table 6.1 shows these events with their corresponding information messages.

Event	message			
	source	target	topic	payload
pre-dispatching at t	$v \in CS$	dispatcher	*state*	$(\texttt{loc}_v^t, \texttt{dest}_v^t, \texttt{seats}_v^t, \texttt{request}_v^t)$
dispatching	dispatcher	$v \in CS$	*schedule*	$\{(r, t_r) : r \in R_v \subset R\}$ R_v: v's assigned requests t_r: potential time to serve r

Table 6.1: Centralized dispatching messages

Once the dispatcher calculates a feasible solution, it sends a *schedule* message to each vehicle v in its CS, the payload consisting of a list of tuples (r, t), each of which specifies a request r assigned to v and the corresponding *pick_up* time t.

6.2 AV-OLRA as an Integer Linear Program

Let a weighted oriented graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \omega \rangle$ defines a one-to-one representation of an urban network, where $\mathcal{N}$ is a set nodes representing city locations, the edge set $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ defines the direct links between nodes, and ω defines the travel time matrix associated with $\mathcal{E}$. ω_{ij} is the travel time associated with edge (i, j) if $(i, j) \in \mathcal{E}$, otherwise it is ∞) as shown in the example illustrated by Figure 6.4 and Table 6.2.

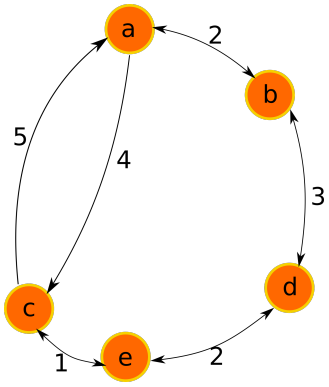


Figure 6.4: Urban network graph $\mathcal{G}$ (example)

	a	b	c	d	e
a	0	2	4	∞	∞
b	2	0	∞	3	∞
c	5	∞	0	∞	1
d	∞	3	∞	0	2
e	∞	∞	1	2	0

Table 6.2: Travel times ω associated with the example of Figure 6.4

6.2.1 Urban Network Pre-processing

The base graph for every AV-OLRA instance is derived from $\mathcal{G}$ by completing it with the shortest paths between nodes. We get a graph $G = \langle N, A, t \rangle$ in which $N = \mathcal{N}$, $A = \{(i,j) : i,j \in N, i \neq j\}$ is a set of arcs between all nodes in N, and $t = \{t_{i,j} : (i,j) \in A\}$ defines the set of shortest-path lengths of arcs in A, i.e. the shortest travel time between two distinct nodes in N. These computations are done offline, i.e. once for the whole urban network before instantiating any scenario. Figure 6.5 illustrate the generation of G from $\mathcal{G}$.

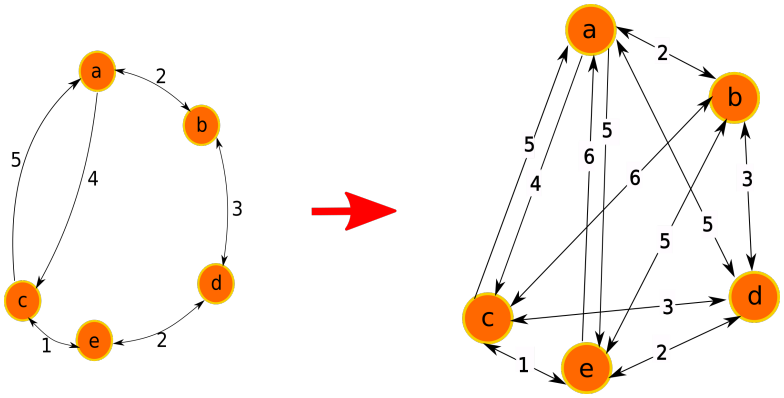


Figure 6.5: Generating the base graph G from the urban network graph $\mathcal{G}$

6.2.2 Formulation

Let us consider n requests to be served and a fleet of k vehicles available to serve them. Each vehicle v has an initial location in the considered scenario to which it must return by the end of its shift. The maximum shift duration, defined by the time horizon T, cannot be exceeded by vehicle routes. A graph $G_s = (N_s, A_s, t_s)$ represents the considered scenario, we define it in the following along with all other notations.

- A node set $N_s = \{1, ..., 2n + k\}$ made up of pick-up nodes $P = \{1, ..., n\}$, corresponding drop-off nodes $d = \{n + 1, ..., 2n\}$, and vehicle initial locations $V = \{2n + 1, ..., 2n + k\}$;
- a set of arcs $A_s = (V \times P) \cup A_{pd} \cup (D \times V)$, where $A_{pd} = A_d \cup A_p$ is the set of pick-up/drop-off arcs, composed of:
  - drop-off arcs $A_d = \{(i, n + i) : i \in P\}$ used by a vehicle during its shift to reach the drop-off spot of a picked request;

  - pick-up arcs $A_p = \{(j,i) : j \in D, i \in P, j \neq i+n\}$ used to go to a new pick-up location for another request after a drop-off;

  the arcs in $V \times P$ (resp. $D \times V$) allow vehicles to go from their starting point to the pick-up spot of the first served request (resp. from the drop-off of the last served request back to their initial locations) at the beginning (resp. at the end) of their shift;

- the edges in A_s are associated with finite running times $t_{i,j} \geq 0$ as rounded-up integer values, and cost proportional to the running time by a factor q. The case $t_{i,j} = 0$ can occur for $(i,j) \in (V \times P) \cup A_p \cup (D \times V)$, whereas for $(i,j) \in A_d$, $t_{i,j}$ is strictly positive.

Further features of the problem being defined: $K = \{1, ..., k\}$ is the set of vehicles, each $k \in K$ being associated with the starting point $2n + k \in V$; a set R of n requests is defined, each $r \in R$ being a couple made up of a unique pick-up node $i(r) \in P$ and a pick-up time window $TW(r) = [l_r, u_r]$, i.e. the time interval in which the client will accept to be picked up: not sooner than l_r and not later than u_r:

$$0 \leq l_r \leq u_r \leq T - (t_{i(r),i(r)+n} + \min_{k \in K} t_{i(r)+n,2n+k})$$

where the upper bound on u_r accounts for the possibility having at least one vehicle capable of serving r before the end of its shift.

For the sake of conciseness, it seems appropriate to introduce an abuse of notation and use symbol i to denote both a pick-up spot and the request in R to which such spot is associated with (instead of r). Consequently, the associated time window is noted $TW_i = [l_i, u_i]$. The min in the bound on u_i becomes a max in case we impose that all vehicles (instead of at least one) should be able to serve client i. The service of a request is not compulsory.

We also define the profit coefficients c_0 and c related to the service of a request: serving request $i \in R$ gives rise to a profit $c_0 + c \cdot t_{i,n+i}$, i.e. the sum of a fixed service fee c_0 and a variable fee proportional by c to the time to reach drop-off from pick-up. Coefficients c_0, c and q are defined in such a way that choosing between serving a request or not is not trivial.

We can now define AV-OLRA as an Integer Linear Program (ILP) by introducing the following decision variables:

- binary variables $x^k_{i,j}$, with $k \in K$ and $(i,j) \in A_{pd}$, and $x^k_{i,j} = 1 \Leftrightarrow$ vehicle k goes from node i to node j;

- binary variables y^k_i, with $k \in K$ and $i \in P \cup D$, and $y^k_i = 1, i \in P \Leftrightarrow$ vehicle k begins its shift from pick-up point i, or $y^k_j = 1, j \in D \Leftrightarrow$ vehicle k ends its shift after drop-off point j;

- integer variables $z_i, i \in P \cup D, z_i =$ time of arrival at pick-up/drop-off node i.

The objective function of the AV-OLRA ILP is the net income Δ, which we aim to maximize:

$$\Delta = \sum_{i \in P, k \in K} (c_0 + c \cdot t_{i,i+n}) \cdot x^k_{i,i+n} - q \cdot \sum_{k \in K} (\sum_{(i,j) \in A_{pd}} t_{i,j} \cdot x^k_{i,j} + \sum_{i \in P} t_{2n+k,i} \cdot y^k_i + \sum_{j \in D} t_{j,2n+k} \cdot y^k_j) \quad (6.1)$$

Therefore, AV-OLRA can be represented by the following ILP model:

$$\max \quad \Delta$$

$$\texttt{s.t.} \quad \sum_{i \in P} y_i^k = \sum_{j \in D} y_j^k \leq 1 \qquad \forall k \in K \tag{6.2}$$

$$\sum_{k \in K} x_{i,n+i}^k \leq 1 \qquad \forall i \in P \tag{6.3}$$

$$y_i^k + \sum_{(j,i) \in A_p} x_{j,i}^k = x_{i,n+i}^k \qquad \forall i \in P, k \in K \tag{6.4}$$

$$x_{j-n,j}^k = \sum_{(j,i) \in A_p} x_{j,i}^x + y_j^k \qquad \forall j \in D, k \in K \tag{6.5}$$

$$\sum_{k \in K} t_{2n+k,i} . y_i^k \leq z_i \qquad \forall i \in P \tag{6.6}$$

$$z_i + t_{i,i+n} \leq z_{i+n} + T \cdot (1 - \sum_{k \in K} x_{i,i+n}^k) \qquad \forall i \in P \tag{6.7}$$

$$z_j + t_{j,i} \leq z_i + T \cdot (1 - \sum_{k \in K} x_{j,i}^k) \qquad \forall (j,i) \in A_p \tag{6.8}$$

$$l_i \leq z_i \leq u_i \qquad \forall i \in P \tag{6.9}$$

$$z_j + t_{j,2n+k} \leq T.(2 - y_j^k) \qquad \forall j \in D, k \in K \tag{6.10}$$

$$x_{i,j}^k \in \{0,1\} \qquad \forall k \in K, (i,j) \in A_{pd} \tag{6.11}$$

$$y_i^k \in \{0,1\} \qquad \forall k \in K, i \in P \cup D \tag{6.12}$$

$$z_i \in \mathbb{Z} \qquad \forall i \in P \cup D \tag{6.13}$$

Constraints (6.2) state that a vehicle could be used or not, but if used, then its shift must begin by visiting some pick-up spot and end after visiting some drop-off spot. Inequalities (6.3) state that a request must be served by at most one vehicle.
Equations (6.4) and (6.5) enforce the fact that if a request is served, then the vehicle visiting its pick-up spot (be it the first of the vehicle shift or not) must be the same that goes to the drop-off spot, as well as the same that leaves it to either serve a new request or end the shift.
Inequalities (6.6) give a lower bound to the arrival time at the pick-up spot of a visited request, forcing it to be not less than the time required for the vehicle assigned to the request to reach it from its starting point, in case the request is the first of the vehicle shift. Based on inequalities (6.7), the arrival time at a drop-off point j is at least the arrival time at the corresponding pick-up spot $j - n$, plus the travel time between the two spots, provided that the request is served by a vehicle, otherwise the relation is loosened.
Similarly, inequalities (6.8) establish a relation between the arrival time at a drop-off spot and that of the (possibly) following pick-up spot visited by the same vehicle. Relations (6.9) force arrival times at pick-up spots to be comprised within the associated time windows, while (6.10) enforce that the arrival time of a vehicle to any of the dropoff spots must leave the vehicle enough time to go back to its starting point before the end of the shift. The domain of the decision variables is defined by (6.11) - (6.13).
The present model is valid under the following assumptions:

- The number of places associated with requests is neglected, i.e. every vehicle can fit every request;

- after pick-up, the only possible next node is the matching drop-off spot, i.e. no overlap of request services.

6.3 Dynamic Rescheduling with Central Dispatcher

When not all requests are known in advance, i.e., new ones can pop out at runtime, the nature of the problem changes, and a dynamic version of AV-OLRA must be designed.

Suppose to have a solution S_0^{opt} consisting of an optimal set of schedules for the requests known at a given date t_0.

Consider now a later date $t_1 > t_0$ in which not all of the schedules of S_0^{opt} have been completed yet, i.e. some requests are still waiting to be picked up or dropped off, and suppose that at t_1 a set of new requests have appeared that were not known at t_0.

The best insertion of those requests in the vehicle schedules of S_0^{opt} cannot guarantee the optimality of the solution. A new solution must then be computed from scratch, considering the whole set of previously known but not yet served requests, and the new ones that have appeared between t_0 and t_1.

6.3.1 Dynamically Computing a New Solution as an AV-OLRA Problem

The AV-OLRA problem as it has been formulated in section 6.2 can come at hand to perform such a dynamic reassignment of requests. We let:

- t_0 and t_1 denote the dates of, respectively, the last computed solution, and that at which a new solution must be computed to accommodate new requests appeared in the meantime;
- R_t denotes the whole set of requests known at a generic t and whose service has not yet begun at t. At any time $\tau > t$, R_t is partitioned into four sets, namely:
  - $R_t^{\tau:w}$, the requests of R_t that are still $w$aiting at τ;
  - $R_t^{\tau:p}$, the requests of R_t that have been $p$icked up, but not yet dropped off, at τ;
  - $R_t^{\tau:r}$, the requests of R_t that have been $r$ejected because impossible to serve, at τ;
  - $R_t^{\tau:c}$ the requests of R_t whose service has been $c$ompleted at τ;
- $R_{t,t'}$ denotes the set of further requests that have appeared at any time τ s.t. $t < \tau \leq t'$.

The dynamic problem of re-optimizing the schedules at time t_1 can be seen as an AV-OLRA in which:

- the request set is $R_{t_1} = R_{t_0}^{t_1:w} \cup R_{t_0,t_1}$. Indeed, requests in $R_{t_0}^{t_1:c}$ and $R_{t_0}^{t_1:r}$ are over, while the service of those in $R_{t_0}^{t_1:p}$ is ongoing and the assignment of a vehicle to them makes no longer the object of a decision
- the vehicles are not required to go back to their starting point after dropping off the last request

6.3.2 Deriving the AV-OLRA Instance Graph G_s

Dynamic re-optimizing can occur multiple times and successive runs will possibly share a part of the spatial information, since they will concern sub-parts of the same city scenario. This is why it is important, at the very beginning and once for all, to trace a 1-to-1 representative graph of the city scenario $\mathcal{G}$, and complete it to obtain a graph G (as it has been explained in section

6.2.1) from which the graph G_s (see section 6.2.2) on which each re-optimization run is based will be derived.

The graph $G_s = (N_s, A_s, t_s)$ on which we define AV-OLRA instances to perform dynamic reassignment is derived from the list of requests and the city scenario graph G:

- pick-up and drop-off spots P and D of node set N_s can be directly obtained from the node set N of graph G, it is sufficient to specify the mappings $N \to P$ and $N \to D$
- subset $A_{pd} = A_d \cup A_p$ of arc set A_s can be directly inferred from the arc set A of graph G
- as for the initial positions of vehicles (nodes V of node set N_s), they are either:
  (a) their starting position if they have never been used, or
  (b) their current position if they are empty, waiting, or, more generally, marauding, or
  (c) somewhere between a pick-up spot and the related drop-off spot, if they are performing a request service at the date t_1 when the dynamic reassignment must be solved.

The third case in particular shows that vehicle initial positions could correspond to any of the nodes of the underlying urban network. This, along with the fact that vehicles do not need to go back to initial positions, suggests defining the node set $V = \{2n+1..2n+k\}$ as a set of dummy nodes and the arc sets $(V \times P)$ and $(D \times V)$ as sets of dummy arcs. The travel time $t_{2n+k,p}$ associated with an arc $(2n+k, p) \in (V \times P)$ are:

- if the initial position of a vehicle at date t_1 matches a node $i' \in N$: the travel time $t_{i',p}$ to reach the considered pick-up spot
- if the initial position of a vehicle at date t_1 does not match any of the nodes in N: the travel time to reach the nearest node $i' \in N$, plus the travel time $t_{i',p}$ to reach the considered pick-up spot. This holds in particular for a vehicle who is performing a request r' at t_1: the travel time will be the residual time to reach the drop-off spot $d' \in N$, plus the travel time $t_{d',p}$.

Note that node d' (as well as node i' in the previous cases) does not need to be included in N_s: only the related travel time need to be taken into account in time $t_{2n+k,p}$ associated with arc $(2n+k, p)$.

One last aspect to be dealt with is how to take into account the fact that vehicles do not need to return to the starting point. This can be modeled as follows:

1) associate travel times equal to 0 to the arcs in $(D \times V)$. By doing so, the last term of objective function (6.1) will be neglected, and the second term at left-hand side in (6.10) will not count, and
2) choose for the time horizon T the value $\max_{i \in P}(u_i + t_{i,i+n}) - t_1$, i.e. the timespan between the latest possible drop-off time and the time t_1 at which the re-optimization solution must be made available

In this case the objective function becomes:

$$\Delta = \sum_{i \in P, k \in K} (c_0 + c \cdot t_{i,i+n}) \cdot x^k_{i,i+n} - q \cdot \sum_{k \in K} (\sum_{(i,j) \in A_{pd}} t_{i,j} \cdot x^k_{i,j} + \sum_{i \in P} t_{2n+k,i} \cdot y^k_i) \qquad (6.14)$$

i.e. the last term is explicitly removed, while inequalities (6.10) become

$$z_j \leq T \cdot (2 - y^k_j) \quad \forall j \in D, k \in K \qquad (6.15)$$

An alternative to Point 1) consists in defining a particular version $G_{cs} = (N_{cs}, A_{cs}, t)$ of graph G_s in which $N_{cs} \equiv N_s$ and $A_{cs} = (V \times P) \cup A_{pd}$, i.e. the part $(D \times V)$ of A_s is explicitly omitted.

Graph G_{cs} is a *bipartite graph* (see Harris et al. [72]), as N_{cs} is partitioned into two sets, namely $V \cup D$ and P, which are stable sets, and the journey of each vehicle in a feasible solution is a bipartite walk in graph G_{cs} starting and ending in $V \cup D$ as shows Figure 6.6.

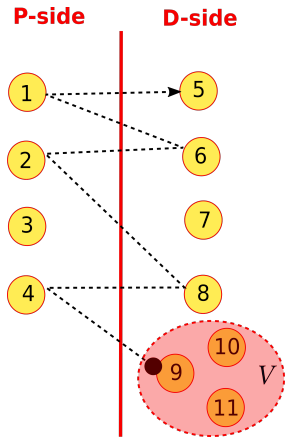


Figure 6.6: Bipartite graph and bipartite walks

Note that in both cases (explicitly removing arcs of $D \times V$ or assigning them null travel times) the concept of "shift duration" makes no sense anymore and the time horizon is kept only for compliance with the previous model. To this end, the timespan between the latest possible drop-off time and t_1 seems the most appropriate choice for T.

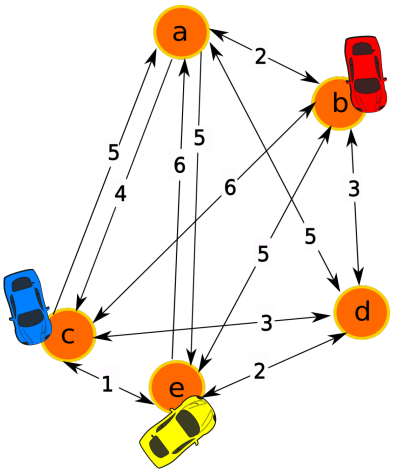


Figure 6.7: Vehicle initial locations

Known requests
$r_1(a \to d,\ [8, 10])$
$r_2(e \to b,\ [9, 11])$
$r_3(c \to d,\ [5, 7])$
$r_4(b \to a,\ [10, 13])$
$r_5(e \to a,\ [7, 9])$

Table 6.3: Request set

Example 1 *Figure 6.7 illustrates a scenario of $k = 3$ vehicles (blue, red, yellow) positioned respectively at (c, b, e) in an urban network represented by the base graph $G = (N, A, t)$ which is the same shown in Figures 6.4 and 6.5. The set of requests consists of $n = 5$ requests $\{r_1, ..., r_5\}$ shown in Table. 6.3, each as a tuple $(o_r \to d_r, tw_r)$. so that $P = \{a, e, c, b, e\}$ $D = \{d, b, d, a, a\}$.*

	P_1 (a)	P_2 (e)	P_3 (c)	P_4 (b)	P_5 (e)	D_1 (d)	D_2 (b)	D_3 (d)	D_4 (a)	D_5 (a)
$P_1(a)$						*5*				
$P_2(e)$							*5*			
$P_3(c)$								*3*		
$P_4(b)$									*2*	
$P_5(e)$										*6*
$D_1(d)$		*2*	*3*	*3*	*2*					
$D_2(b)$	*2*		*6*	*0*	*5*					
$D_3(d)$	*5*	*2*		*3*	*2*					
$D_4(a)$	*0*	*5*	*4*		*5*					
$D_5(a)$	*0*	*5*	*4*	*2*						
$V_1(c)$	*5*	*1*	*0*	*6*	*1*					
$V_2(b)$	*2*	*5*	*6*	*0*	*5*					
$V_3(e)$	*6*	*0*	*1*	*5*	*0*					

Table 6.4: Matrix representation of $G_{cs}(N_{cs}, E_{cs}, t_{cs})$

The spatial constraints of this instance can be encoded in a bipartite graph $G_{cs}(N_{cs}, E_{cs}, t_{cs})$ *consisting of* $2n + k = 13$ *nodes connected by a set of edges whose weights* t *are shown in Table 6.4.*

Summary

In this chapter we described the centralized dispatching approach to solve *AV-OLRA*. With this approach, a central dispatcher is associated with every connected set to solve dynamic instances of the problem. We discussed in detail through this chapter the dispatcher agent selection methods, information gathering and message types, the *ILP* representation of *AV-OLRA* static instances, and finally the necessary change to the initial *ILP* in order to deal of with the dynamic reassignment.

Chapter 7

Decentralized Solutions

In Chapter 4, we presented our generic model (*AV-OLRA*) for the ODT allocation problem within a dynamic context. We discussed the communication architecture and the functional specifications of the centralized approach to solve *AV-OLRA* in Chapter 6.

The problem at hand is considered as dynamic in both spatial and temporal dimensions. In the temporal dimension, the requests are not known in advance but announced in a stochastic manner at run-time. Moreover, the knowledge of the vehicles is also limited to the information shared with its connected set: this implies the dynamics in spatial dimension as the vehicle location affects the context of the problem. This dynamics may affect the quality of upcoming planning decisions dramatically and reduces the benefits of the already taken planning decisions.

In this chapter, we explore the direction of *decentralized* solutions in which vehicles are considered as autonomous agents and can interact in peer-to-peer manner to coordinate their local decisions to avoid the potential conflicts. We have defined in Section 5.2 the sub-behaviors of the AV agent and argued that the implementation of the planning sub-behavior is what defines the solution method, which is characterized by the adopted coordination mechanism $CM = (DA, AC, AM)$. Hence, all the approaches presented in what follows, have in common the decision autonomy attribute pointing to *decentralized allocation* ($DA = D$), while they differ in their level of information sharing and cooperativeness (AC), and the allocation mechanism (AM) that each agent follows to build his local solution.

7.1 Selfish Decentralized Allocation

Being a *selfish* agent signifies that it does not care about the preferences of other agents when making a decision; i.e., it may take them into account, but finally, what dictates its choice between possible alternative decisions are the benefits that these decisions bring to it. Moreover, the decisions taken at moment t are those bringing maximum direct benefits for the agent concerning only the problem context at t no matter what the context will be at an upcoming moment $t' > t$.

For our problem, the set of available options for a selfish AV agent is its set of known requests if the agent has no passenger on board, otherwise no option is considered. Because of that, the state *Coordinating* of the planning sub-behavior is ignored (as if it reaches an agreement for any chosen option), so that the quality of the solution is dependent on the ranking and priority

functions of the agents for the upcoming requests. In general we may refer to *greedy* algorithms to implement allocation methods for this kind of approaches.

Among the most recent examples of selfish approach is the work of van Lon et al. [155]. When a vehicle is not already carrying customers, it has to decide which request it will handle first, depending on the information it has about available requests. A heuristic computes a priority value for each request. Then, the agent handles the request with the highest priority value first. Conflicts can arise, but are solved simply by applying the first arrival policy.

7.1.1 Specifications of Greedy Algorithms

The term "greedy" defines not only one but a family of algorithms that make the locally optimal choice at each stage without ever calling int into question during following stages. The choices that a greedy algorithm makes may rely on the choices made so far, but not on future choices. This is the main difference with the other alternatives such as *Backtracking*, that incrementally builds candidates to the solutions, and abandons ("backtracks") a candidate as soon as it determines that the candidate cannot possibly be completed to a valid solution or to a solution that is better than the currently best known one.

Using greedy approaches is common in many resource allocation problems, and not only in transport. For example, when a space in a homeless shelter becomes available, the agency may offer it to the household ranked as being in the highest need; when deceased donor livers become available, they are offered first to those who are medically matched and with the highest measure of need (e.g. MELD[1] scores).

So far, what identifies a greedy algorithm is the priority function, which chooses at each step the best candidate to be added to the solution, i.e. a function that assign scores to the available options so the algorithm can pick the option with the highest score.
The challenge here lies in the design or choice of such a function where only local and incomplete information are available. The work of van Lon et al. addresses this problem by using genetic programming[2] to compute the fitness of individual priority functions for vehicle agents to solve dynamic DARP instances and automatically choose appropriate ones.

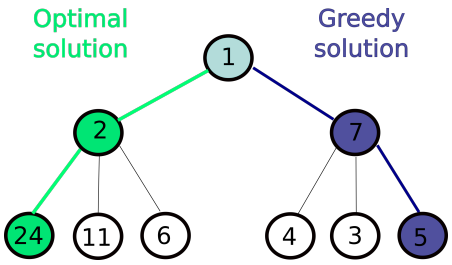


Figure 7.1: Cases of failure of greedy algorithms

Greedy algorithms usually fail to find the overall optimal solution because they usually do not operate exhaustively on all the data. They may commit to certain choices too early, which prevents them from finding the best global solution later, as in Figure 7.1. In this simple example we have a decision tree in which each path to a leaf node from the root defines the set of decision steps (nodes) to reach a feasible solution. The profits of each individual decision is shown by the label for the corresponding node. A solution to the problem here is to choose, starting from the root node, a path to a leaf node maximizing the sum of profits at each step. The greedy

[1] The Model for End-Stage Liver Disease, or MELD, is a scoring system for assessing the severity of chronic liver disease

[2] Genetic programming is an evolutionary approach that works by defining a goal in the form of a quality criterion (or fitness) and then using this criterion to evolve a set (population) of candidate solutions (individuals) [156].

algorithm will choose what appears to be the best immediate choice, so it will choose 7 instead of 2, and will not reach the best solution, which is 26.

However, the ability of greedy algorithms to quickly find feasible solutions makes them more scalable, and hense very popular for solving several real-world, real-time problems which typically feature dynamic settings and online decision-making process.

7.1.2 Greedy Heuristic to Solve AV-OLRA

Recalling the AV agent planning sub-behavior of AV-OLRA model defined in Section 5.2.3, having a greedy behavior means that agents do not rely on each other and never exchange their plans information ($AC = N$). The allocation mechanism to this model is a greedy-based one ($AM =$ Greedy), in which the vehicle may consider only one request in advance (e.g., the closest one to shorten the empty driving distance); to gain the most immediate utility, without taking into account future arrivals.

The heuristic we consider as an example of the selfish approach in this work is based on the results obtained by van Lon et al. under the assumption that vehicle agents can schedule only one request in advance (considering dynamic DARPs, long term planning is not beneficial because of the rapidly changing dynamics). In this approach, when a vehicle is not already carrying passengers, it has to decide which request it is going to handle first depending on information it has about available requests.

Decision Mechanism

In the following we make use of the notations introduced in Sections 4.3 and 5.1. The set of available options for a vehicle agent v_i at a time t is:

- The set $\mathcal{R}^t_{v_i}$ of, unsatisfied, available requests, known by v_i at t, if v_i is not carrying a passenger on board.

$$\mathcal{R}^t_{v_i} = \{r \in \texttt{known}^t_{v_i} : st^t_r \in \{\texttt{announced, available}\}\}$$

- The empty set Φ if the vehicle is carrying a passenger, and hence cannot schedule requests.

A priority function is used by the vehicle agent to compute a priority value for each request. Then, the agent handles the request with the highest priority value first. The priority values for all available requests are recomputed every time tick, and agents do not share information about request priorities.

We consider that the vehicle agents are homogeneous. Therefore, they use the same formula for the priority function. Given a vehicle v, the priority function returns for a request r a value that is inversely proportional to the cost of selecting the r to be the next request for v.

$$\texttt{priority}^t_v : \mathcal{R}^t_{v_i} \rightarrow [0, 1]$$

$$\texttt{priority}^t_v(r) = \frac{1}{cost^t_v(r)}$$

Thus, the lower the cost is, the higher the priority the resource has. In its simplest form, the cost function returns at a moment t the distance of reaching o_r the pickup location of r from the location $\texttt{loc}^t_v$ of v, thus agents assign the highest priority to the request with the nearest pick-up location, with respect to their current location. Figure 7.2 illustrates an instance of AV-OLRA with three vehicles (V1, V2, V3) aware of two requests (R1, R2). The weight of each edge stand

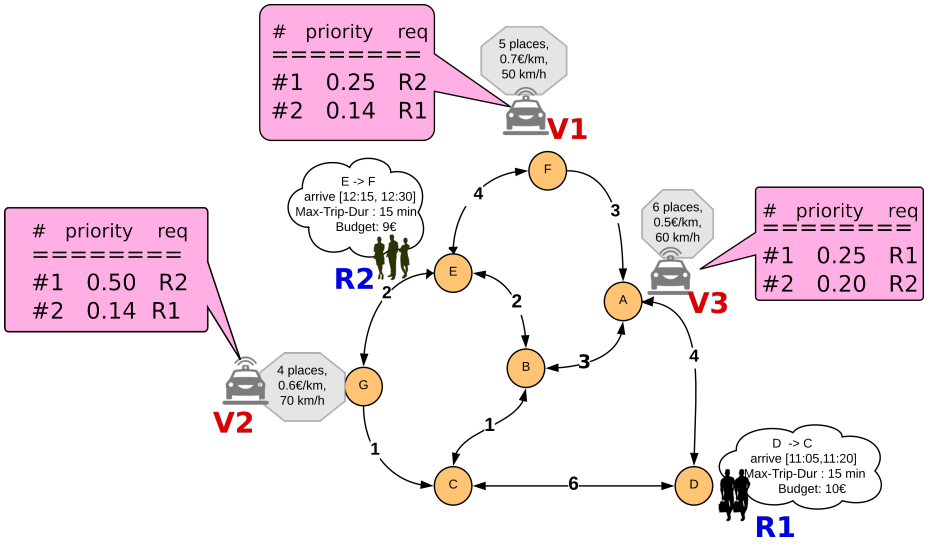


Figure 7.2: A scenario with 2 requests and 3 greedy vehicles using the closest-request priority

for the average trip duration. Every vehicle sorts its list of known requests by the descending priority based on the closest request priority function (considering the temporal distance). As a consequence, V1 and V2 prioritize R2, while V3 prioritizes R1.

Conflict Resolution

Working in a shared environment, each agent would like its own interests to be given the most consideration. Unfortunately, the interests agents may overlap, leading to the conflicts. In our scenarios, conflicts arise when two or more vehicles are interested in the same request. The selfish behavior implies that vehicles do not coordinate their decisions by sharing information about their schedules, and conflicts cannot be avoided. However, they could be handled by applying the first arrival policy. i.e. the vehicle who arrives first to the pick-up location takes the request.

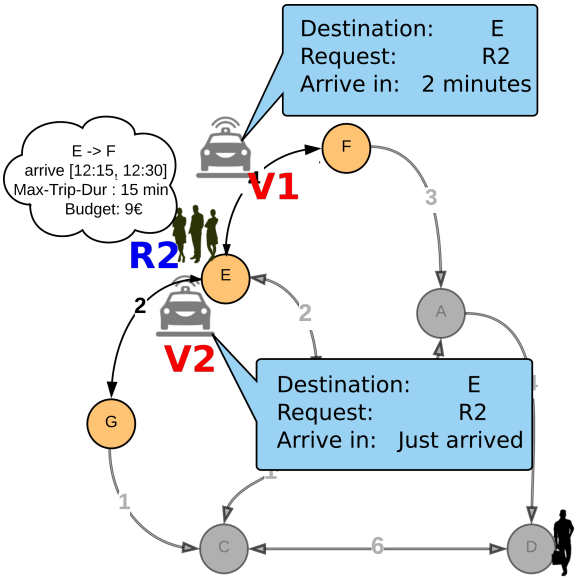


Figure 7.3: Conflict resolution with first arrival policy

Recalling the previous example, two vehicles V1 and V2 were prioritizing R2 at t_0, accordingly

they both leave towards E. At $t_0 + 2$, One of them (V2) arrives at E while the other is midway between F and E as shows Figure 7.3. Applying the first arrival policy, V2 handles R2 that disappears from the list of available requests, while V1 is forced to choose another one. The drawback here is the ignorance of what lies ahead of the taken decisions and the fact that conflicts are not resolved until a very late stage, which reduces the QoB because of the operational cost of the frustrated trips, such as for V1 in our example.

One possible way to reduce this loss in QoB is to bring forward the detection of the conflict, by establishing commitments that change the status of requests when they are chosen by a vehicle (a blackboard coordination) as done by Danassis et al. [40]. To handle this in our model we need to give vehicles an access to a shard memory in which they can store information about vehicle decisions, or make them announce their schedules via message exchange without violating the specifications of the selfish behavior. However, this requires additional computation and verification of the context data validity and trustworthiness of messages in such a competitive environment.

7.2 Coordination-based Approaches

As with human beings, autonomous agents performing in challenging domains can potentially achieve better results by working in teams than by working alone. Ideally, agents will coordinate to reallocate resources among them in a way that allows them to achieve their goals in a reliable and efficient manner. Coordination can lead to faster task execution, increased robustness, higher quality solutions, and the accomplishment of goals that are unachievable for a single agent. In this case, the coordination mechanism is decentralized ($DA = D$), the agents share messages to coordinate their decisions ($AC = S$), and a coordination protocol is applied by the allocation process. There exist several approaches to implement such a coordination mechanism like market-based approaches [45, 50, 2] and distributed constraint optimization (DCOP) [56].

7.2.1 Market-Based Coordination

Market-based multiagent coordination approaches have received a great deal of attention and are gaining popularity in both Economics and Computer Science. With increasingly sophisticated market economies, humans have had to deal with coordination challenges for thousands of years. In such economies, individuals and self-interested communities exchange goods and services to maximize their own profit; such reallocation of goods and services results in more benefits to the system as a whole [45].

Market-based coordination is one of the effective and proven ways to resolve conflicts in distributed systems [39]. In the last fifty years, researchers have applied the principles of market economies to multiagent coordination. The earliest examples of market-based multiagent coordination that appeared in the literature are the work of Farber and Larson [55] and the Contract Net protocol by Smith [146]. In market-based multiagent systems, agents are conceived as selfish actors operating in a virtual economy. The goals to achieve and the resources available are valued commodities that can be exchanged. For example, in this domain, trip requests can be assigned to vehicle agents via market mechanisms such as auctions. When an agent completes a trip, it receives some sort of reward in the form of virtual income. However, the agent must also be charged for the time and resources it consumed to serve the request, i.e. the operational cost of the trip. The essence of market-based approaches is that, in a well-designed system, the process of exchanging tasks and resources between agents to maximize individual profit simultaneously improves global efficiency of the team.

The rest of this section consists in a general overview of the most known market-based multiagent coordination paradigms, describing their main features in addition to the general architec-

ture of a market-based mechanism, while we explain our contributed auction-based coordination mechanism in Chapter 8.

Auction Theory

Since ancient times, auctions have been used to answer the most fundamental questions in Economics: who should get the goods and at what price?
A common aspect of auctions is that they receive information, in the form of *bids*, from potential participants about their willingness to pay the price, and the result –i.e., who wins what and how much has to be paid– is determined exclusively based on the information received. This implies that auctions are universal, in the sense that they can be used for many scenarios to determine winners [86], especially in resource allocation [28, 144].
Given a set of bids B in a combinatorial auction on a set of resources R, as shows Figure 7.4, the winner determination problem (WDP) is defined as finding an allocation of items to bidders that maximizes the auctioneer's revenue [92]. The WDP is known to be an NP-hard problem because each subset of bids would have to be checked for feasibility, and the revenue that this subset of bids provides, computed and they are in a number that is exponential in the number of bidders. In its simple form, when the auction is about a single, indivisible resource $r \in R$, the question becomes, "to whom r should be allocated?", which can be expressed as:

$$winner : R \rightarrow B \cup \phi$$

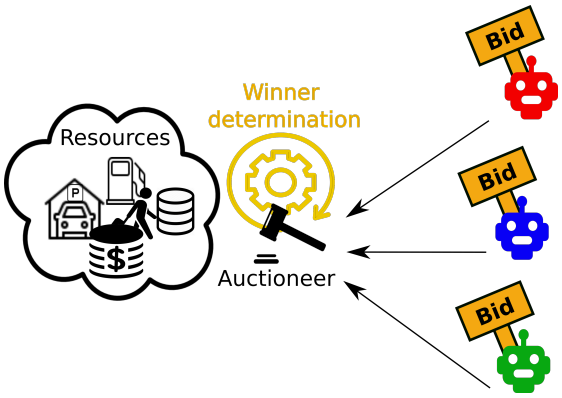


Figure 7.4: General architecture of an auction system

Negotiation

Negotiation is a topic of interest in multiagent research, likewise in Economics and Political Science. Negotiation can be used to address conflicts in a wide variety of multiagent domains [80]. Usually, a negotiation-based conflict resolution framework for multiagent systems, along with agents and their actions in dynamic environments, involves a set of negotiation strategies and a negotiation protocol.Agents must propose offers consistent with their preferences. If they reject the opponent's offer, they must propose a counter-offer.

A negotiation strategy is a policy for generating the next offer. It helps the negotiator to decide whether to concede or offer an alternative, i.e. a settlement. Each agent has knowledge of what the world could be (usually incomplete knowledge). A negotiation strategy is chosen by an agent based on its knowledge. A negotiation protocol is the set of rules that governs the interactions during a negotiation session (also called "thread"). It covers the possible participant

types, negotiation states, events that can change the negotiation states, and the valid actions of negotiators.

A variety of negotiation protocols have been proposed in the literature of Economics and Computer science. Many of them have been implemented for multiagent conflict resolution, the most common being the Contract Net [146], the alternative offer [115], the monotonic concession [48], and the one-step negotiation [131] protocols.

Market Equilibrium

In market-based system, an agent is considered to be acting competitively when he makes an offer, ignoring the impact of his own action on the market's equilibrium price. Under the assumption of perfect competition, the constrained optimization problem of each agent is parameterized by the cost and utility functions. We say that an agent is in equilibrium if its current set of offers matches the outcome of its optimal solution of the problem. If all agents and all resources are in equilibrium, the allocation of resources determined by the auction results is a competitive equilibrium. A *competitive equilibrium* has several desirable properties from a mechanism design perspective, including the two fundamental welfare theorems of general equilibrium theory:

1. All competitive equilibria are Pareto optimal, as no agent can do better without another doing worse.
2. Any Pareto optimal allocation of resources represents a competitive equilibrium for some initial endowment.

Competitive equilibria provably exist and can be computed [158], for example, by algorithms based on fixed-point methods [139] and gradient-based optimization techniques [163]. However, by using equilibrium equations, these methods violate the decentralization concerns that motivate the current problem.

7.2.2 Distributed Constraint Optimization Approach

In this class of coordination mechanisms, agents exchange information and cooperate to achieve a common objective, avoiding conflicts while optimizing the solution quality. Constraint reasoning techniques help to ensure that the dynamic knowledge that resides in system entities solves the resource allocation optimization problem without imposing unrealistic requirements that all agents continuously communicate their local knowledge to a traditional optimization solver.
The constraint reasoning generally refers to Constraint Satisfaction Problem (CSP) and Constraint Optimization Problem (COP). CSP represent the entities in a problem as a homogeneous finite collection of constraints over variables, which is solved by constraint satisfaction methods. In general, constraint satisfaction is NP-complete and constraints are generally expressed as binary constraints. Distribution can be considered as an extension of classical centralized CSP where each agent is responsible for assigning one or several variables with relative autonomy. Distributed Constraint Optimizartion Problems (DCOPs) [166] can be expressed as constraint graphs, in which the vertices are the variables and the edges are the constraints. DCOPs represent problems in which agents coordinate their allocations of values in a decentralized way in order to optimize their own objective functions. Even if it does not have a global vision, every agent can communicate with his neighbors in the constraints graph. DCOPs typically address the issue of reaching a global optimum given the interaction graph of a set of agents. This approach can be used to model and solve efficiently a diverse range of problems. Problem solving and communication strategies are closely related within DCOP. This feature allows the computational components of a DCOP to use the interaction graph structure of the agents to generate powerful solutions [56].

7.2.3 DCOP Model for AV-OLRA

In a DCOP approach to the AV-OLRA problem which is the subject of our study, agents could decide on their own but coordinate with agents of the same connected set (CS) using a distributed constraint optimization algorithm to avoid conflicts within the connected set.
Once an agent detects a change in the context of the problem (e.g. a change in set of known requests or in number of agents in the CS), this change should be reported by broadcasting a message to all members of its CS. Upon receiving this message, all agents broadcast their current location, destination, available seats and current request on board if any. Once this information is exchanged, a DCOP $\big(A, X, D, C, \varphi\big)$ is generated from the AV-OLRA instance to maximize objective function in Equation 4.24, as follows:

- A defines the set of agents in the connected set
- X defines the set of decision variables in three subsets (x_{ij}^v's, y_r^v's and z_r^v's):
  - x_{ij}^v is a binary variable that takes 0 as a value if vehicle v moves on an arc (i, j),
  - y_r^v is a binary variable that takes 1 if and only if the request r is the first request to be served by v,
  - finally, z_r^v is an integer variable defining at what time a request r is visited by v,
- D defines the domains of variables:
  - binary $\{0, 1\}$ for x_{ij}^v and y_r^v, $\forall v, i, j, r$
  - a set domains defining the time-window range $[l_j, u_j]$ for each z_i^v,
- C defines the set of constraints, which consists of hard constraints (spatio-temporal availability, and time windows) similar to what is expressed in relations (6.2) -(6.10), and soft constraints defining the cost and utility of the allocation decision (used to calculate the value of the objective function).
  In this model we have mainly two functions that are used to calculate the values of soft constraints:
  - **loss function**, used to compute the operational cost of the movement of a vehicle v from an origin node i to a destination node j

    $$\texttt{loss}(v, i, j) = -cpd_v \cdot \texttt{dist}(i, j) \cdot x_{ij}^v$$

    where $\texttt{dist}(i, j)$ is the trip distance from i to j.
  - **profit function**, used to compute a positive value defining the individual profit of a vehicle v from serving a request r, i.e. when $x_{o_r,d_r}^v = 1$

    $$\texttt{profit}(v, r) = (P + p \cdot \texttt{dist}(o_r, d_r)) \cdot x_{o_r,d_r}^v$$

    where P is the service fee (fixed price) and p is the coefficient for price per distance.
- φ defines the partition of variables from X to agents from A. In the most straightforward way, every vehicle agent is considered responsible for the variables that directly concerns it. i.e. an agent v' is responsible of the set of variables $x_{i,j}^{v'}$, $y_r^{v'}$ and $z_r^{v'}$ $\forall i, j, r$

Thus, at each time a change is detected in the context of a CS, a new DCOP is is generated representing the sub-problem instance. Once agents receive their local versions of the DCOP (their subsets of variables, domains and constraints), the adopted algorithm is triggered, and agents start exchanging messages accordingly until reaching a solution or detecting the impossibility to solve the DCOP instance at hand. In the first case, each agent update its schedule with the new allocations. Agents keep from their old schedules only the trips that have already started, as they were not been part of the DCOP instance. In the other case, agents keep their entire old schedules and continue their journeys as planned until a new scheduling round.

Summary

In this chapter, we explored the direction of decentralized approaches to solve the ODT allocation problem. We overviewed these approaches from te coordination mechanism (CM) point of view of the planning sub-behavior of AV-OLRA agents.
In this family of solution methods, the coordination mechanism is based on the decision autonomy of agents to produce a decentralized allocation ($DA = D$). We classified these approaches based on the cooperativeness level (AC) of agents, and more precisely into: *Selfish* in which agents do not rely on each other and never exchange their plans ($AC = N$); and *Coordination-based* approaches ($AC = S$) where a coordination protocol is applied by the allocation process. Based on the applied coordination and the implemented allocation method AM, the coordination-based solutions are categorized as competitive and cooperative families: the first family is that of Market-based approaches, while the second family is that of DCOP-based approaches. For each of these families, we explained the features, requirements and solution architecture. In the next chapter we explain in details our contributed decentralized, market-based heuristic (ORNInA) for efficiently solving AV-OLRA instance in dynamic settings with runtime optimization. The implementation details for centralized dispatching and the different decentralized approaches including *Selfish*, *ORNInA*, and *DCOP* are presented in Chapter 9, then experimental evaluation and result analysis of these approaches are shown in Chapter 10.

Chapter 8

Combining Dynamic responsiveness Together with Solution Quality Refinement

According to Bellini et al. [14], ODT optimization should be able - as its most prominent features - to:

- yield transportation solutions that are as much as possible door-to-door solutions;
- minimize waiting times, walking paths, and vehicle changes.

The computational complexity of ODT allocation problems makes it difficult to a central dispatcher to handle real-world size instances of ODT problem, and the dynamics of the problem during execution (online requests, variable fleet size, traffic problems, and other environment dynamics) increase the difficulty of the task. Therefore one may expect that decentralization can be an appropriate solution to these problems.

In the context of ODT, we proposed in Chapter 4 the AV-OLRA problem model, illustrated in Figure 8.1. In Chapter 7 we explored the direction of decentralized solutions and coordination mechanisms. We notably overviewed the features of market-based coordination mechanisms.

In what follows, we propose *ORNInA*, a new market-based decentralized heuristic for the *AV-OLRA*. ORNInA benefits from the fast responsiveness of insertion heuristics and the good results gained by k-opt optimization techniques.

Figure 8.1: A sample AV-OLRA problem instance, with demand sources (triangles) and taxis (circles) with their respective communication ranges (in blue).

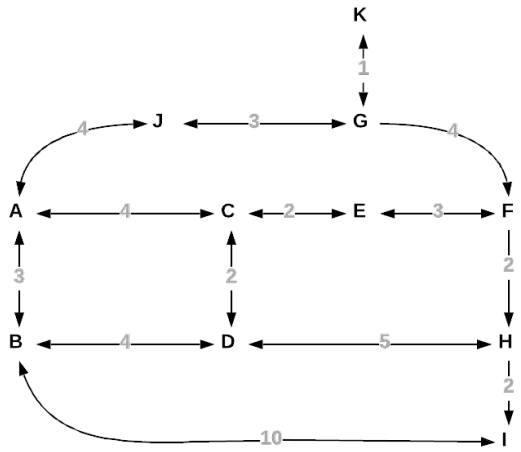


Figure 8.2: The road network of a sample AV-OLRA instance

8.1 Auction-based Insertion Heuristics

ORNInA stands for "**O**nline **R**escheduling with **N**eighborhood exchange, based on **In**sertion heuristic and **A**uctions". This section presents our new insertion heuristic, and how vehicles coordinate their decisions through auctions. The main objective of vehicle agents is to provide feasible schedules that maximize their utility by minimizing the global operational cost of serving the maximum number of requests.

Given a vehicle v having a potential demand set D_v, providing a schedule for v that satisfies all the requests $d \in D_v$ means solving a routing sub-problem to produce the best ordering of requests in the schedule. Considering the dynamic aspects of our model, we use an insertion heuristic like the one described by Solomon [147] to adapt local vehicle schedules continuously. This type of insertion heuristics aims to select requests whose insertion costs minimize a measure of total route distance and time (so called *Marginal Cost*). The result of this algorithm is a set of requests; each of them is associated with the potential time at which a vehicle will be at the pick-up location.

Insertion heuristics are a wildly used method for solving a variety of scheduling and routing problems, as they allow to quickly find a feasible solution even if with no guarantee on solution quality, as it is the case for every heuristic approach. For instance, in Vehicle Routing Problem (VRP), solutions are created by repeatedly inserting unscheduled demands in a partially constructed route or as the first demand in a new route. Local search, e.g. with the k-exchange neighborhoods (k-opt), can then improve the co-constructed solution. Insertion heuristics are proven to be efficient in finding feasible schedules very fast [22], even though, since the scheduling of requests is handled one by one, performances depend on the number and the order of demands.

To cope with this limitation, an improvement phase based on k-opt can be applied. k-opt is a path improvement algorithm, at each iteration, k segments from the current plan are replaced by k segments to get a cheaper path [73].

8.1.1 Priority Function

In our model, the agents handle the new requests based on their priority order. An agent uses a priority function to assign priority values to the requests he knows. Given a vehicle v, the priority function, similar to the *greedy* one (see Section 7.1.2) computes at time t for a request

r a value that is inversely proportional to the marginal cost of inserting r in the schedule of v:

$$\texttt{priority}_v^t(r) = \frac{1}{\texttt{cost}_v^t(r)}$$

Thus, the higher priority is given to the new request with lower insertion cost. The cost value $\texttt{cost}_v^t(r)$ is time dependent, i.e. it is computed at a moment of time t by a vehicle agent v depending on its beliefs on the current problem context that could change in the future. In its simplest form, the cost function returns the distance of the pick-up location of d from the location of v. Thus agents assign the highest priority to the request with the nearest pick-up location w.r.t. their current location.

Each agent determines his schedule to maximize the measure of quality of its solution. Since several vehicles may be interested in the same request, we need a coordination mechanism to resolve conflicts. Here we will use an auction-based mechanism for this purpose.

8.1.2 Agents Bid Criterion

In our model, we use a *first-price auction form* [86] to answer the question "which vehicle will consume the request and at what cost?". When a vehicle v becomes aware of a request r, it ranks it in its queue according to the priority it has assigned to it. At the time t, v selects a request $r_{selected}$ which is the first in the queue, generates a set of alternatives, each of which is a potential schedule resulting from the insertion of $r_{selected}$ in a feasible step (that does not violate any of $r_{selected}$ constraints) of its current schedule. Every alternative is associated with a *cost*, which is the marginal operational cost of adding this request to the schedule. The choice with the best *cost* is considered to broadcast an offer:

$$Bid_v^{r_{selected}}(t_{start}, cost)$$

with t_{start} the time of pickup for $r_{selected}$. Listing 1 presents the pseudo code of the functions to update the request priority queue and selecting the highest priority request to bid for it.

Listing 1 Priority update and bidding for requests

```
Function updatePriorities():
    priorities ⟵ [ ];
    foreach r ∈ knownRequests do
        if myBids.getBid(r) == Null then
            p= computePriority(r);
            priorities.insert(< p, r >)
        end
    end
    if priorities.size() > 0 then
        bid(priorities.getFirst().getRequest())
    end
End Function
Function bid(Request r):
    c = computeCost(r);
    t = computePickupTime(r);
    sendBidMessage(r,c,t);
End Function
```

In this work, we consider the operational cost of trips as a linear relation to their lengths, so that for a vehicle whose $cpd = 1$, the cost of a trip is simply its total length (sum of the distances, as shown in Figure 8.2). We consider the value of $cpd_v = 1$ for every vehicle in the following examples. Therefore, the marginal cost of insertion is the difference in path length between the initial path and the new path.

8.1.3 Winner Determination in ORNInA

Recalling the winner determination formalism presented in Section 7.2.1, the question "which bidder vehicle wins the request $r \in R$ that is a single, indivisible, unsharable resource?" can be expressed as a linear relation. Seeking decentralization, we introduce a binary version of this relation corresponding to our problem

$$win : V \times R \rightarrow \{0, 1\}$$

where the bidders are the vehicles and the resource items are the requests. In this decentralized winner determination mechanism, every vehicle $v \in V$ that is aware of a request r is responsible for determining the value of $win(v, r)$ and thus whether to insert r (if $win(v, r) = 1$) in its schedule or not as shows Listing 2.

We assume all agents to be truthful collaborating agents. The default value for $win(v, r)$ is 0. After their announcement, the bids remain available for a specific time period t_{expire}. Until t_{expire} the vehicle listen to bids of other vehicles on r, if it receives a better offer, it assigns the value 0 to $win(v, r)$ which means it determines itself as a looser and withdraws from the auction. On t_{expire} if the bid cost of v is less than any other bid $Bid^r_{v'}$ received at $t_{Bid^r_v} + t_{expire}$ to serve a request r, v considers itself the winner of the auction (i.e. assigns 1 to $win(v, r)$), and updates its schedule with the new bid path. This mechanism implies that on t_{expire} of each auction, r is allocated to at most one vehicle (in the CS).

Listing 2 Decentralized winner determination

```
Function updateSchedule(Request r):
    currentSchedule.insert(r);
End Function
Function handleBidExpiry(Bid b):
    if (b.isTimeOut()) then
        updateSchedule(b.request);
        mybids.remove(b);
    end
End Function
Function evaluateBid(Bid b):
    b1= myBids.getBid(b.request);
    if b1 ≠ Null then
        if b1.cost < b.cost then
            broadcast(b1);
        else
            mark_unfeasible(b1);
        end
    end
End Function
```

8.2 Online Solution Improvement in ORNInA

The dynamics of AV-OLRA dynamics may affect the quality of upcoming planning decisions dramatically and reduces benefits of the already taken long-term planning decisions [155]. In the following, we illustrate some examples of the effects of environment's dynamics on the performance of the insertion heuristic and then propose a local optimization protocol to improve the quality of the solution.

Example 2 *The simple scenario in Figure 8.3 shows two vehicles V_1 and V_2 located in A and B, with empty schedules at the beginning. Considering the time is sampled as a series of ticks*

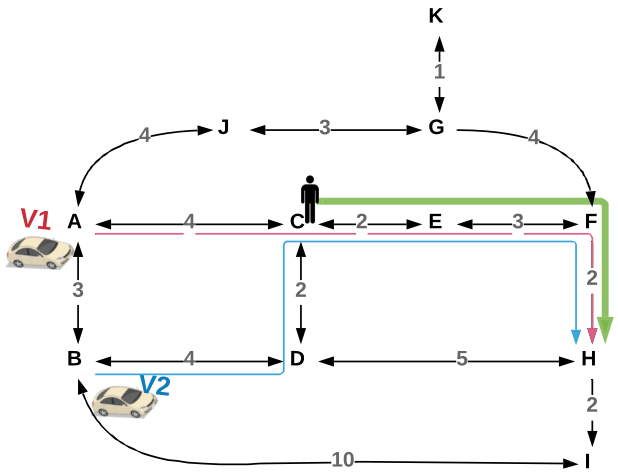


Figure 8.3: V_1 wins the auction to serve d_1 from C to H

$\{t_0, t_1, t_2, ...\}$, and the weights of arcs represent trip duration in ticks (temporal distance). At tick t_1, the first request d_1 is announced: $d_1 :< C, H, [t_{10}, t_{30}], 1,$ "announced" $>$. Both vehicles now know d_1. V_1 can serve it by following the path $A \rightarrow C \rightarrow E \rightarrow F \rightarrow H$, so V_1 places the offer $Bid_{V_1}^{d_1}(t_{10}, 11)$. V_2 can serve it via the path $B \rightarrow D \rightarrow C \rightarrow E \rightarrow F \rightarrow H$, so issues the offer $Bid_{V_2}^{d_1}(t_{10}, 13)$. Note that both vehicles can reach C earlier than t_{10}. However, to comply with the request constraint, the potential pickup time of the bid should belong to the request time window whose lower-bound is t_{10}. When V_2 receives $Bid_{V_1}^{d_1}$ it determines itself as a looser by setting $win(V_2, d_1) = 0$, while receiving $Bid_{V_2}^{d_1}$ by V_1 will not change anything for V_1. Assume the bids are available only for one tick duration, at t_2 the vehicle V_1 stops listening to other vehicle bids on d_1, considers itself as the winner, and adds d_1 to its schedule, so that the overall operational cost of the fleet is now 11.

8.2.1 Dynamic Settings

The problem at hand is considered as dynamic in both spatial and temporal dimensions. An example of these dynamics is the stochastic announcement of requests and being unknown in advance as shows Example 3.

Example 3 *Figure 8.4 shows a situation where the use of the bid-based insertion heuristic guarantees the reactivity, but not the achievement of the best possible scheduling.*
After bidding on d_1 at t_1 and adding it to V_1 at t_2 as shown in the previous example, a new request $d_2 :< J, K, [t_{15}, t_{40}], 1,$ "announced" $>$ arrives at t_3 and both vehicles become aware of it and place their possible bids. In the absence of any exchange capacity, V_1 still has d_1 in its schedule (with an initial cost of 11), so the best offer it can place is to serve both requests (d_2 then d_1) with a marginal cost of 14. While V_2 places the winning bid $Bid_{V_2}^{d_2}(t_{15}, 11)$, it adds d_2 to its schedule, and the overall cost becomes 22. Note that in this case, serving d_2 with V_1 and letting V_2 take care of d_1 (as shown in Figure 8.5) results in an overall gain of global operational cost which becomes 21, but this solution is never realized because d_1 is already scheduled on V_1.

To be able to make effective bids for new requests or improve the solution, we also propose that the vehicles exchange their planned requests, as illustrated in Figure 8.5. In the following, we propose a local optimization protocol to improve the quality of the solution for similar cases.

This protocol is based mainly on a relocation heuristic [125] that replaces k segments from the vehicle schedule by k segments from the schedule of a neighbor vehicle. This heuristic works

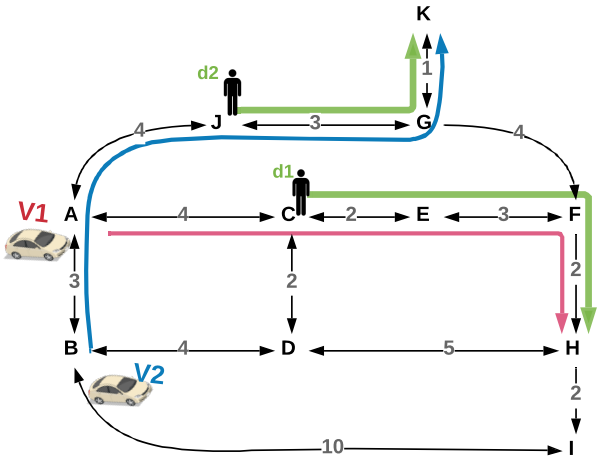


Figure 8.4: With no demand exchange, V_2 wins d_2 and V_1 keeps d_1

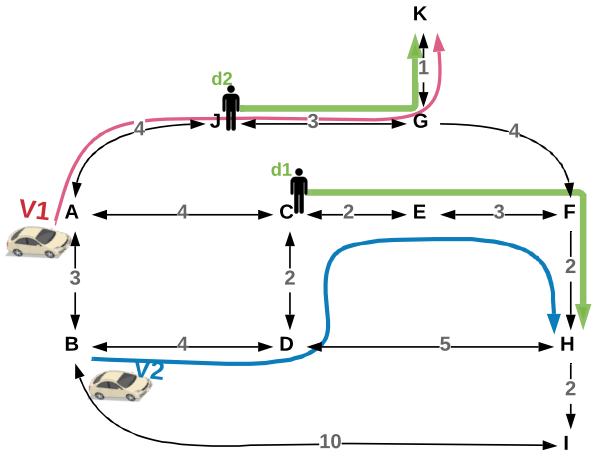


Figure 8.5: Global improvement when V_1 abandons d_1 to serve d_2

in a similar manner of the k-opt that relocates segments in the same vehicle route in order to optimize its operational cost, while the difference is that we consider the relocation happens between two routes of different vehicles. In its simplest settings, k is set to 1, so we introduce $1-reloc$, in which for each iteration, at most one demand can be relocated from a vehicle schedule to another. In this work, we seek the implementation of the simple auction mechanism 1-reloc to avoid dealing with the NP-complete winner determination of the combinatorial auctions. However, if vehicles want to exchange more than one requests they are still able to do so by several iterations, each is 1-reloc.

Example 4 *Let's consider another case shown in Figure 8.6, where V_1 has d_1 in its schedule, V_2 has empty schedule, and a new demand is announced $d' :< (t_{10}, t_{20}), D, F >$. V_1 offers two alternative bids $Bid_{V_1}^{d_1,d'}((t_{10}, t_{12}), +4)$, and $Bid_{V_1}^{d'}(t_{10}, +15)$ (taking into account abandoning d_1), while V_2 can only offer $Bid_{V_2}^{d'}(t_{10}, +13)$.*

Obviously $Bid_{V_1}^{d_1,d'}$ is the winner and V_1 takes the charge of both demands with global cost $+15$, while the optimal solution in this case is that V_2 who takes both demands which make the global cost only $+13$, but using the auctions with abandon strategies lets V_2 in this case either bid for d' alone when it is announced, or bid for d_1 alone in response of the abandon suggestion by V_1. So in this case, the optimal solution is not achievable with the aforementioned abandon strategies.

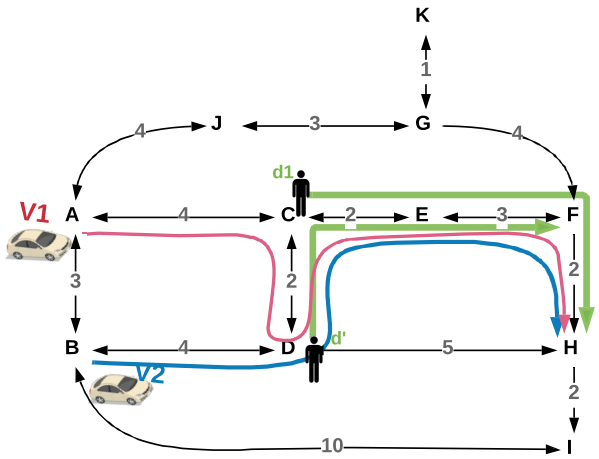


Figure 8.6: While it can serve both demands, V_2 can only bid for one demand at a time

8.2.2 Pull-demand Optimization Bids

In this section, we introduce the *pull-demand* protocol; an optimization protocol to improve our heuristics, which is similar to some extent to what Agatz et al. [2] proposed as the *rolling horizon* strategy with few differences.

In the *rolling horizon* strategy, all the schedules of vehicles are considered temporary, until they are below a temporal threshold called the *planning horizon*, and thus any request is available to be scheduled by any vehicle, unless it is considered as a *committed request* by particular events; for example, a vehicle v has started serving (moving towards) a demand d, whose upper-bound of time-window becomes below the planning horizon threshold that is rolled forward periodically at runtime.

The application of this strategy requires that all vehicle schedules information are shared, so that when a vehicle v_i offers to serve a request d, it knows if it is scheduled by another vehicle v_j, and therefore if it should send its offer cost to v_j. Then v_j will calculate its estimation to the global gain (or loss) in operating cost by abandoning d compared to the cost proposed by v_i. If there is a gain, it agrees to abandon d and then v_i updates its schedule with d, otherwise the bid is rejected. In our protocol, we do not use the concept of committed request or the planning horizon, but a vehicle can determine, based on its believes, whether it can satisfy a request or not; and thus it can only bid on requests that it can satisfy, so requests that are rescheduled or that do not have enough time to be rescheduled are automatically ignored by the agent.

Another difference is that we don't have access to a shared memory. Agents exchange information about the environment and requests through information messages. Moreover, in the work of Agatz et al., optimization is performed periodically at a predefined frequency [2], while the protocol we propose is executed in parallel with the auction-based insertion strategy to have a fast rescheduling for continuous announcement of requests. Based on shared information of the current context, the optimization protocol is executed in each connected set of vehicles when any change in the set is detected (the set of vehicles in the connected set is changed) or at least one of them becomes aware of some requests already scheduled by others. This strategy is thus performed as a protocol of five steps as follows:

Step 1 A set of new demands enters the system sorted according to their announcement time.

Step 2 Each new request is distributed to the connected set to which the sources (see Section 5.1.1) belong. Each agent in this set can select among new requests, and scheduled and unscheduled requests that have not yet reached their scheduled departure time.

Step 3 Each agent enters the auction process to serve demands it has chosen in *Step 2* in similar auction criteria of that we explain in Section 8.1.2.

Step 4 Each agent searches among its scheduled requests for the one to satisfy the next tick. Let us assume that the list of potential requests is not empty, and let d_{next} be the one chosen to be satisfied in the next tick. The agent broadcasts a message to inform other agents that it handles d_{next}, and each receiver deletes it from its list of potential requests.

Step 5 Each agent deletes any request that reaches its time window upper-bound because staying any longer available for rescheduling would violate its time constraints.

Step 6 The scheduled and unscheduled requests that still have time remain announced by their sources (*Step 2*). This allows better planning in the next tick if new requests are announced or some new agents join the connected set.

This protocol, performed by each agent to comply with ORNInA coordination mechanism, can be represented by a state machine as shows Figure 8.7. The agent moves from one state to another based on inner events, time constraints or on receiving messages from other agents.

Listing 3 Bid criteria and pull-demand auctions

```
Function improveCandidates():
    improvementCandidates ⟵ [ ];                          ▷ State Improving Candidates
    foreach v ∈ CS do
        foreach r ∈ v.schedule do
            g=computeGain(r);
            if g ≥ 0 then
                improvementCandidates.insert(< g, r, v >)
            end
        end
    end
    if improvementCandidates.size() > 0 then
        pullBid(improvementCandidates.getFirst())     ▷ select the best pull-demand candidate
    end
End Function

Function updatePriorities()          ▷ first check winner determination for previous bids:
    foreach b ∈ myBids : b is not marked "unfeasible" do
        handleBidAnswer(b,null)
    end
    priorities ⟵ [ ];                                          ▷ sort requests by priority
    foreach r ∈ knownRequests do
        if myBids.getBid(r) == Null then
            p= computePriority(r);
            priorities.insert(< p, r >)
        end
    end
    if priorities.size() > 0 then
        bid(priorities.getFirst())                        ▷ select the highest priority request
    end
End Function

Function bid(float priority, Request r)          ▷ State Bidding, requires Listing 6:
    c,t = computeCostAndTime(r);
    sendBidMessage(r,c,t);
End Function

Function pullBid(float gain, Request r, Vehicle owner)   ▷ State Bidding, requires Listing 6:
    c,t = computeCostAndTime(r);
    sendPullDemand(r, c, t, owner.id);
End Function
```

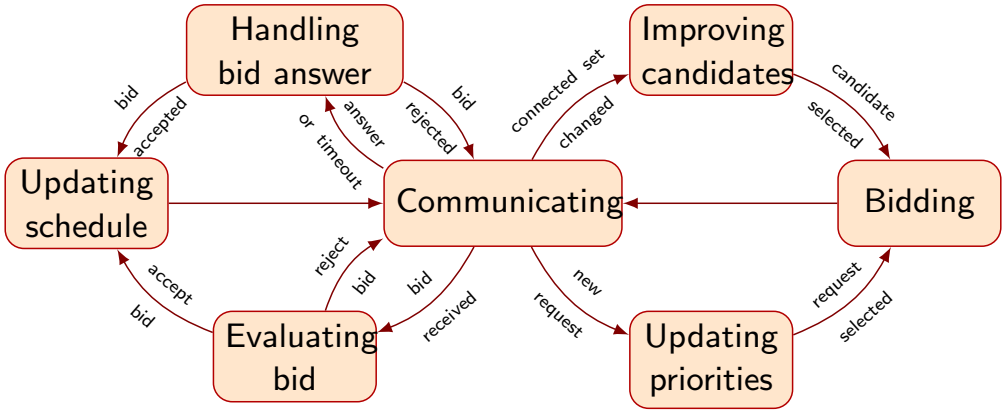


Figure 8.7: ORNInA coordination mechanism

Listing 3 extends Listing 1 with the *pull-demand* auctions (Function `pullBid(gain, request, owner)`) whose rescheduling candidates are determined by the Function `improveCandidates()`.

Note that in this 1-reloc, at each time, only the highest gain candidate becomes a subject of exchange. The decentralized winner determination mechanism is also extended to handle the cases of pull-demand auction in addition to the basic bids. Listing 4 illustrates this mechanism.

Listing 4 Winner determination

Input: Schedule currentSchedule, List myBids ▷ Requires: Listing 5, Listing 6

```
Function evaluateBid(Bid b, int bidderId):
    if b is Pull-Bid then
        l= computeLoss(b.r);
        if l < b.gain then
            m=Message(this, "bid_answer", bidderId, {bid=b, answer="accept"});
            m.send();                          ▷ Message(senderId,topic,receiverId,payload)
            abandon(b.r)                                              ▷ accept the exchange
        else
            m=Message(this, "bid_answer", bidderId, {bid=b, answer="reject"});
            m.send();                                                 ▷ reject the exchange
        end
    else
        b1= myBids.getBid(b.request);          ▷ look for its own bid on the same request
        if b1 ≠ Null then
            if b1.cost < b.cost then
                broadcast(b1);                    ▷ reject b and remind others about b1
            else
                mark_unfeasible(b1);              ▷ determines its loss and withdraw
            end
        end
    end
End Function

Function handleBidAnswer(Bid b, Message m):
    if (b.isTimeOut()) or (m.payload.answer=="accept") then
        updateSchedule(b.request);
        mybids.remove(b);
    end
End Function
```

Finally, the basic methods of schedule update and message handling that are necessary to implement these algorithms are illustrated by Listings 5 and 6.

Listing 5 Schedule update

Input: Schedule currentSchedule, List myBids

Function `updateSchedule(`*Request r*`):`
| currentSchedule.insert(r);
End Function

Function `abandon(`*Request r*`):`
| currentSchedule.delete(r);
End Function

Listing 6 Communication in ORNInA

Input: knownRequests priorities mybids improvementCandidates
Function `readMessage(`*Message m*`):`
| **switch** *m.topic* **do**
| | **case** *"new_req"* **do**
| | | knownRequests.add(m.payload);
| | | updatePriorities();
| | **end**
| | **case** *"join"* **do**
| | | improveCandidates()
| | **end**
| | **case** *"known"* **do**
| | | knownRequests.addAll(m.payload);
| | | updatePriorities();
| | | improve();
| | **end**
| | **case** *"bid"* **do**
| | | evaluateBid(m.payload)
| | **end**
| | **case** *"pull"* **do**
| | | evaluateBid(m.payload)
| | **end**
| | **case** *"bid_answer"* **do**
| | | handleBidAnswer(m.payload.bid, m.payload.answer)
| | **end**
| **end**
End Function

Function `sendBidMessage(`*Request r, float cost, Time t*`):`
| m= Message(sender=this,topic="bid", receiver="all", payload=Bid(*r*,*cost*));
| myBids.add(Bid(*r*,*cost*, *t*), currentTime+defaultBidAvailability);
| m.broadcast();
End Function

Function `sendPullDemand(`*Request r, float cost, Time t, int ownerId*`):`
| m= Message(sender=this,topic="pull", receiver=ownerId, payload=PullBid(*r*,*cost*));
| myBids.add(PullBid(*r*,*cost*,*t*), currentTime+defaultBidAvailability);
| m.send();
End Function

8.2.3 Discussion

Given the decentralized context, the insertion heuristic is very efficient in terms of response time. The temporal complexity of the basic insertion heuristic for the Vehicle Routing Problem (VRP) is of $\mathcal{O}(n^3)$[22]. This type of heuristic is often used to solve dynamic DARPs, where new incoming requests have to be continuously processed in real-time and integrated into the evolving schedules of vehicles. The usage of other local search heuristics (e.g. k-opt or relocate) may give local improvements to the solution returned by the insertion heuristic based on the current context. However, the Pull-demand protocol can significantly improve the quality of the solution for other values of k rather than 1, regardless of the complexity of dealing with combinatorial auctions , as illustrated in the following example.

Example 5 *Let us consider again the case of Example 4 illustrated in Figure 8.6, The bids of the vehicles are thus* $Bid_{V_1}^{d'}(t_{12}, +4)$*, and* $Bid_{V_2}^{d'}(t_{10}, +13)$*.* $Bid_{V_1}^{d_1,d'}$ *is the winner, and* V_1 *will handle the two requests with an overall cost of* 15*. With the Pull-demand protocol,* d_1 *and* d' *enters in the set of candidate requests for* V_1 *and* V_2*, so that the vehicles can make combinatorial offers:* $Bid_{V_1}^{d_1,d}((t_{10}, t_{12}), 0)$ *and* $Bid_{V_2}^{d',d_1}(-2)$*. The cost of* V_2 *is* 13 *and the gain of* V_1 *is* 15*).* V_1 *has nothing to change in its schedule.* V_2 *wins and the solution is improved with an additional optimization round. Let's look at the applied protocol, step by step.*

Step 1: d' *enters the system at* t_2 *and both vehicles are aware of this,* V_1 *wins the auction with an overall cost of* 15

Step 2: d_1 *and* d' *are now in the set of requests known by both vehicles,* V_2 *calculates the costs to serve* d_1 *alone (*13*),* d' *alone (*9*) and both requests together making* 13*. It then selects the two requests as its potential requests.* V_1 *has no potential request because it already has the two requests in its schedule.*

Step 3: V_2 *places a bid* $Pull\_Bid_{V_2}^{d',d_1}((t_{10}, t_{12}), 13)$*. For* V_1 *the cost to serve both requests is* 15 *so it accepts* $Pull\_Bid_{V_2}^{d',d_1}$ *because it causes a gain of* 2*.*

Step 4: None of the requests reach their scheduled service time, thus none of the agents select a d_{next}*.*

Step 5: None of the requests reach the upper-bound of their time window.

Step 6: All known requests remain announced and available for the next potential improvement.

Summary

In this chapter, we proposed ORNInA, a decentralized coordination mechanism for the exchange of requests, based on an insertion heuristic and auctions, to allocate requests to vehicles in the context of dynamic on-demand transport with V2V communication. We model this allocation problem with connectivity constraints as an AV-OLRA instance.

The difference from the previously detailed mechanisms is that the agents act competitively, but share information about their decisions to pursue their common objectives, and follow market-based protocols to achieve agreements.

Here the allocation mechanism is auction-based ($AM = Auctions$), bids and rescheduling messages are exchanged between agents ($AC = S$) to coordinate in a peer-to-peer manner about the decentralized scheduling decisions of their fleet of autonomous vehicles ($DA = D$).

This mechanism is proposed to perform in dynamic settings, between vehicle agents that belong to a connected set in which they can receive and send direct or broadcast messages.

Agents interested in some request initiate first-price auctions for it, and the winner adds it to its schedule; the winner determination is completely decentralized process. To improve the scheduling efficiency in dynamic settings, agents are allowed to exchange their scheduled requests at run-time, with additional auction rounds (called "pull-demand" bids) to decide if this exchange increase the value of the objective function within the connected set.

We showed through examples that the request exchange protocol can be a promising improvement in the quality of the solutions. In the following chapters, we describe the implementation details and provide experimental analysis on the performance of this heuristic comparing it to other solution methods.

Part IV

Experiments

Chapter 9

Experimental Framework

The ODT problem has several variants. Several solution methods exist for the same problem, and their quality depends on the definition of the problem and the properties of its instances (e.g., demand, road network).

The engineering of dynamic systems of transport and logistic is a difficult task, simulators are often used during development. Simulation environments are non-trivial software with complex requirements: time management, event management, scalability and configurability for experiments. Advances in multiagent technology have prompted researchers to build simulation frameworks, either as ad-hoc implementations [42, 58, 70], or using generic platforms (e.g., Repast Simphony [117], Netlogo [116]) in which transportation characteristics must be coded. On the other hand, there are MAS platforms dedicated to transportation problems, often mainly focused on traffic (e.g. Sumo [13], Movsim [151], Matsim [8], MITSimLab [15]) in which ODT specifications (strategy, basic vehicle behavior, etc.) have to be implemented. Because of their emphasis on modification, one can use open-source frameworks to address the specification of problems that could be unique to his interests. Some notable open-source multiagent ODT simulators exist such as RinSim [155], Agents4ITS Mobility testbed [170], and TaxiSim [27] in which communication management, communication constraints, and various solution models must be coded.

In this chapter, we propose the design and software engineering behind a simulation framework namely *AV-SIM* with specific ODT tools (transportation data processing, demand generation), a multiagent model with common vehicle behaviors, communication management, and application integration to consider MAS alternatives and specific coordination libraries. This architecture supports the multiagent approach to AV-OLRA model (see Chapter 4).

9.1 AV-SIM Testbed

An important feature of the AV-SIM is its explicit separation of problem and solution. This design decision simplifies development of the multiagent solution methods since it forces designers to separate these two aspects and thus focus on the planning sub-behavior of autonomous vehicle agents. The AV-SIM architecture is implemented for the traffic and transport simulation component of the WebGIS *Plateforme Territoire*[1] (see Figure 9.1), providing tools for designing problem instances (e.g. road network and request definitions), visualizing solving process

[1] https://territoire.emse.fr/

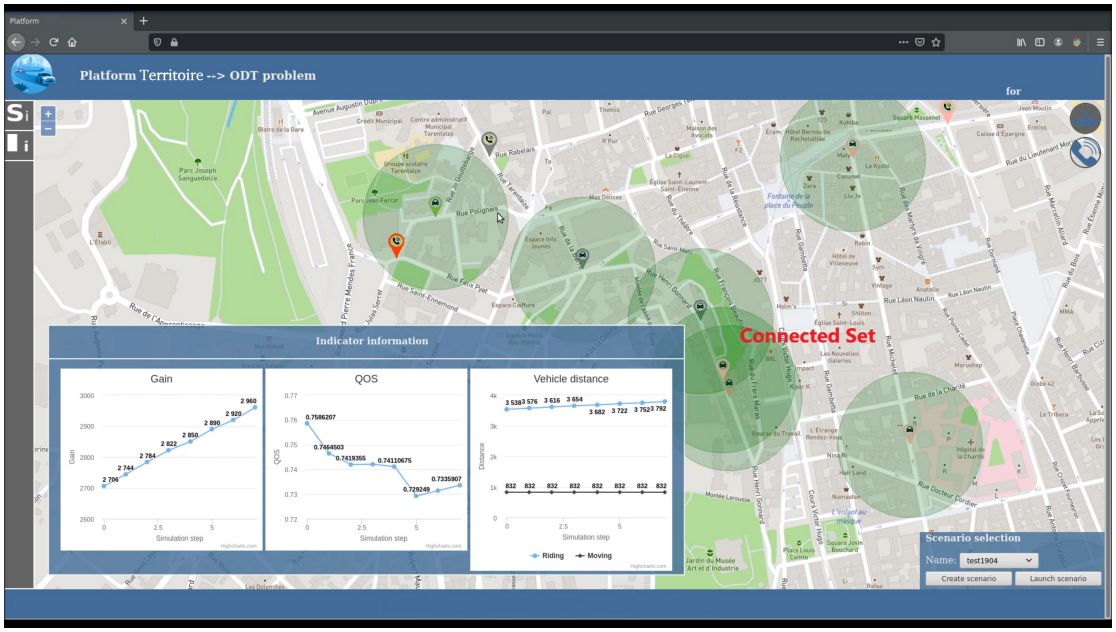


Figure 9.1: A screenshot of the ODT simulator of the territoire platform, showing an AV-OLRA instance with clients (phone) and taxis (vehicle) with their communication ranges (in green).

execution (e.g., vehicle behaviors and states) and results (indicators for assessment).

Territoire platform supports the design of integrated solutions for territorial decision problems. Indeed, a solution may integrate environment, building, mobility, and logistic expertise. This platform aggregates dynamic or static data coming from sensors, web, or local resources.

9.1.1 Framework Architecture

Territoire is a Java-based service-oriented platform, and a solution method is a composition of REST services belonging or not to the platform with specific applications. To foster reusability and solve the interoperability problem between services, *Territoire* services are described following a common knowledge model, and input/output data of the services are in JSON. The architecture of the AV-SIM is represented by Figure 9.2. We consider ODT solutions are based on three types of components:

- generic data processing services for the extraction of transportation network data and specific services for filtering, cleaning, and computing the biggest connected component for a specific geographical zone;
- dedicated, mobility-related or logistic algorithms like Dijkstra or sample vehicle routing algorithms;
- dedicated mobility simulation engine.

It is inspired by the four-step travel model [110]; a service or an application is associated with each step of this model

1. *Trip Generation:* the flow quantification and simulation setting: Network(s) parameters, multiagent parameters, demand parameters, i.e. the scenario of the simulation.
2. *Trip Distribution:* the trips' creation, i.e., the triples (origin, destination, temporal window) definition according to the scenario (synthetic/real data).

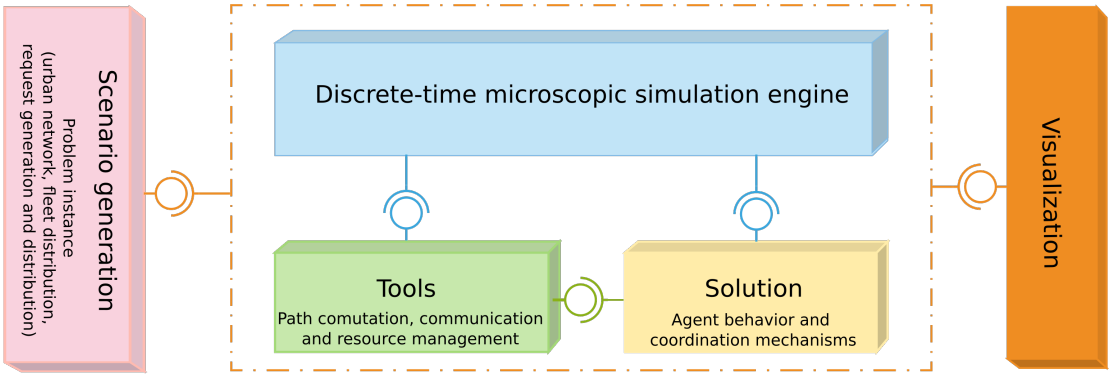


Figure 9.2: Components of the AV-SIM testbed

3. *Modal Choice:* the choice of the mobility mode. For ODT, it corresponds to the route computation by agents in the simulator.

4. *Route Assignment:* the simulation process. It is composed by the allocation process of the vehicles to the requests and the vehicle trip execution.

The two first steps are offline and correspond to the simulation scenario generation, while the next two are online and concern the mobility simulator, which has three main features.

The first is the *management of communication* between simulated elements (client and vehicle) using local communication (VANET) or not. To do this, the simulator considers the concept of connected sets (CS). The composition of these CS changes with the vehicles' movement (see Chapter 4), and the simulator manages the CS's dynamics. All sent messages in a CS are received by all vehicles in this CS. Global communication corresponds to the management of a unique CS.

The second feature is *management of the demand*. The appearance/disappearance of the clients' requests is managed according to the demand definition, i.e., the request origin's spatial-temporal parameters for client appearance and the temporal window or request satisfaction for the client disappearance.

The third is the *microscopic multiagent simulation* of the autonomous vehicles (see next section). In Figure 9.1, there are 3 requests, 7 vehicles, with 5 CS (one with 3 vehicles is identified).

9.1.2 Implementing the MAS Approach to AV-OLRA with AV-SIM

In this section we explain the implementation requirements for our multiagent system presented in Chapter 5 using AV-SIM framework. This MAS is used to simulate and assess solution methods to the AV-OLRA problem. An autonomous vehicle agent (AV), the only type of agent in our model, is associated with each vehicle in the system. The behavior of the vehicle has to consider two activities:

1. activities related to the execution of trips;

2. activities related to the allocation process.

There are three different sub-behaviors (see Section 5.2). First, the Acting sub-behavior is common to all agents and represents the AV life-cycle as a transport vehicle that can pick-up/drop-off

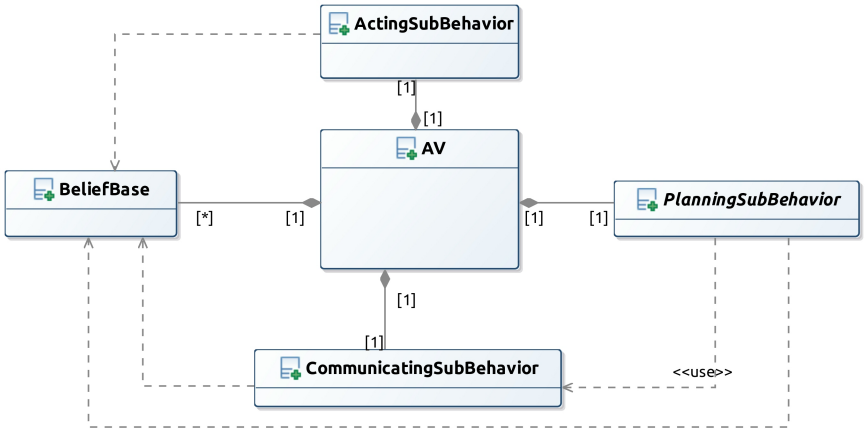


Figure 9.3: Abstraction of AV agent class diagram

passengers, move and stop. These actions affect the spatial status of the vehicle and its availability beliefs. Second, the Communicating sub-behavior is common too (the communicated information are specific to the allocation process). When a message is received and read by an agent, it is stored in its belief base. This message may concern shared information between vehicles and/or a coordination protocol. Finally, the Planning sub-behavior represents how an AV obtains its dynamic schedule in run-time to serve its requests, affecting both spatial and temporal beliefs. This behavior depends on the allocation process.

From an object-oriented point of view, an agent can be seen as an instance of the AV class which is composed of an instance of BeliefBase class and 3 sub-behavior objects, which are instances of ActingSubBehavior, CommunicatingSubBehavior, and PlanningSubBehavior as shows Figure 9.3.

Agent's Belief base

The agent's belief base represents what the agent knows about himself and the context of its surrounding environment. The BeliefBase class represents this knowledge as a dictionary that maps properties (keys) to their values; a tuple $< \texttt{key}, \texttt{value} >$ is a belief. It has also methods to add, remove and update these beliefs. The essential beliefs of a vehicle agents are:

- status: a list of vehicle status properties (seats, location, etc.)
- schedule: a list of tuples $< request, time >$ defining the set of potential requests to serve associated with their potential pick-up times.
- known: the list of all known, unexpired requests.
- connected: the list of vehicles that are member of the current connected set.

Acting Sub-behavior

As a transport entity, an autonomous vehicle should be able to move across the physical environment and interact with its components. It must have functionalities to goTo a destination node, stop moving, and pickUp or dropOff passengers. The acting sub-behavior represents these actions in the form of a finite state machine (FSM) as shows Figure 5.4 on Page 49.

For each experimental scenario, every AV agent has access through its ActingSubBehavior to the urban network instantiated for this scenario. The urban network is an instance of the class

Network and is composed of a set of Source objects where the Request objects have their origins and destinations. The Network class provide functions to compute the paths between sources and determine traffic states of these paths.

The ActingSubBehavior class disposes functions to handle acting events, thus the transitions in the FSM and updating the state of the vehicle. These functions update also the AV's belief base with the vehicle location, state, requests on board and availability.

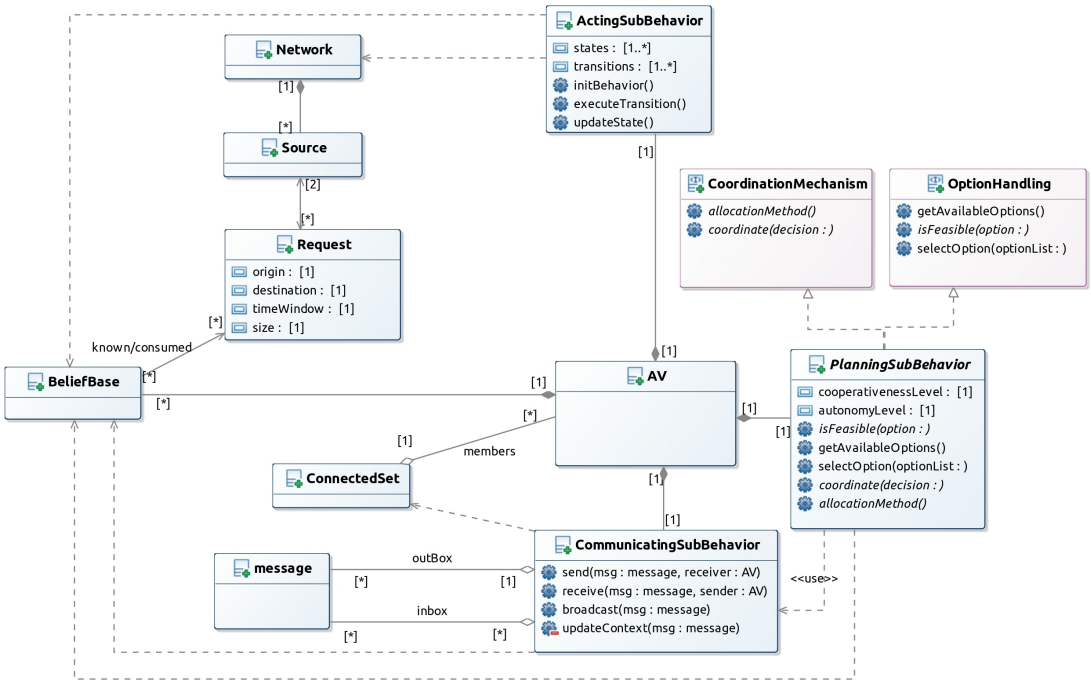


Figure 9.4: AV agent class diagram

Communicating Sub-behavior

Being a communicating agent, an AV should be able to send/receive messages and update its belief with the information included in these messages. Considering the properties and constraints of DSRC (e.g. VANET) an AV can only broadcast messages to vehicle located in its communication range. To increase the connectivity by applying the concept of connected sets, once a vehicle receives a message for the first time, it broadcasts it again to apply the transitive connectivity and ensure the receiving of the message by the maximum number of vehicles.

A message is an instance of class Message. It has a header and body, the header contains the `senderId`, the message `topic`, and optionally a specific `destination`. Depending on the message `topic` the receiving agent determines how to update its knowledge base with the information included in the message body. By default, a message should concern every agent in the connected set. Assigning a value to the `destination` attribute specifies the *id* of the agent who should handle it. Thus, an agent updates its belief base only with messages that have either its *id* or *NULL* as a value for the `destination` attribute. The CommunicatingSubBehavior class implements this protocol through `send`, `receive`, `broadcast`, and `updateContext` methods.

Planning Sub-behavior

The decision autonomy of an AV implies its ability to take planning decisions to build its schedule (i.e. to decide which requests it will serve and at what time). In Chapter 5, we explained that the implementation of the planning sub-behavior depends basically on the adopted solution method which is characterized by the coordination mechanism $CM =< DA, AC, AM >$. This sub-behavior can be abstracted by the FSM shown in Figure 5.5 on Page 51.

The PlanningSubBehavior class is an abstract class that is inherited from extending classes that encapsulate the different solution methods. This class implements two interfaces. The first is the OptionHandling interface that groups methods related to looking for available scheduling options, checking their feasibility and thus selecting a feasible one. Implementing these methods is what define the value of DA. The second interface is the CoordinationMechanism whose methods are used to specify the adopted allocation method (defines AM value) and how to coordinate the planning decisions with other agents, i.e. the message types that should be exchanged during the coordination rules that dictates reaching an agreement or not (defines AC value).

To apply a solution (say `solutionX`) to AV-OLRA as a multiagent coordination mechanism, one should have a class SolutionXPlanningSubBehavior that extends PlanningSubBehavior implementing all its abstract methods. Then assign an instance of this class to the AV agents as a planning sub-behavior (See Figures 9.4 and 9.5).

The AV agent and the details of its sub-behaviors are illustrated in Figure 9.4. It shows also the representation of the spatial and communicational environment: spatial environment is represented by the Network and Source classes while the communicational environment is represented by the ConnectedSet. In the next section, we describe the implementation specification of the planning sub-behaviors for the different methods we used in our experimental scenarios.

9.2 Implementing Solution Methods

Based on the different values of DA, AC, and AM we can obtain a variety of coordination mechanisms. In Chapter 5 we classified these mechanisms into centralized $DA = C$ and decentralized $DA = D$, and shown that decentralized mechanisms can base on individual decisions with information sharing $AC = S$ (thus coordinated decisions) or not ($AC = N$) while AM stands for the allocation method that is used to build the agents' schedules. We have defined 4 categories of solution methods and described in details the functional properties of methods belonging to each of these categories in Chapters 6, 7 and 8. In this section, we explain the implementation guidelines of these methods that we'll use in our experiments using the AV-SIM architecture.

9.2.1 Centralized Dispatching Based on ILP

To implement the dispatching mechanism described in Chapter 6, it is necessary to have a DispatchingPlanningSubBehavior class that extends the PlanningSubBehavior and implements its abstract methods (see Figure 9.5). We consider the *Lower_ID* strategy for the dispatcher selection. Each vehicle has a belief entry `dipatcherId` that identifies the connected set dispatcher. When instantiated, a vehicle v consider itself as the dispatcher. Once messages are received from other vehicles, v becomes aware of their ids and thus it assigns the minimum id to `dipatcherId` value. When a vehicle detects its loneliness in a connected set, it rests the `dipatcherId` value to its own id.
The boolean function isDispatcher() encapsulate the process of determining whether the vehicle consider itself as a dispatcher or not.

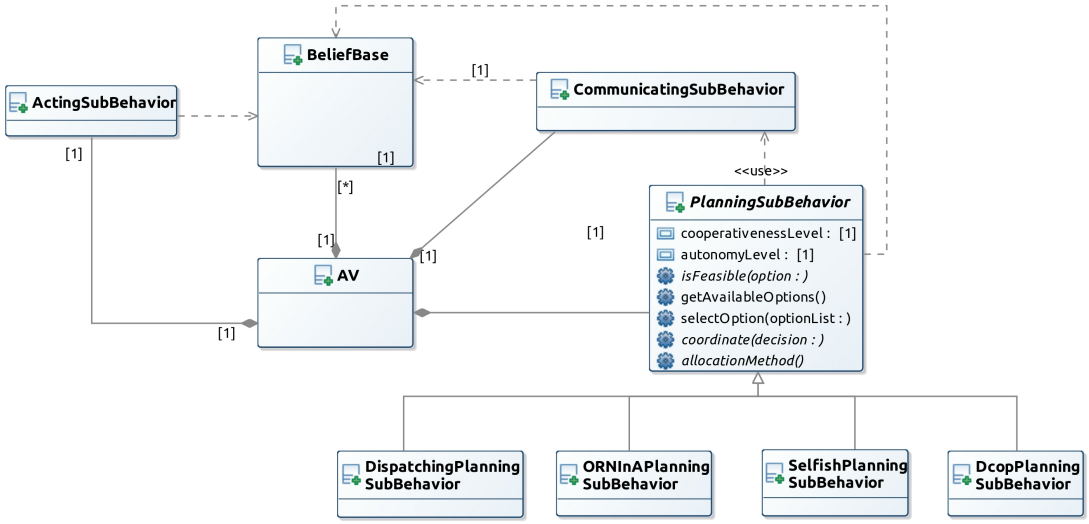


Figure 9.5: Implementations of planning sub-behavior

Planning Options

An AV that has dispatching-based planning sub-behavior can choose at each time cycle its planning decision from a set of 3 options:

- *update_schedule*: this option is only feasible when the vehicle receives a message of topic schedule. It is then the only available option. When chosen the schedule of the vehicle should be replaced by the one received in the message body.

- *compute_solution*: this option is only feasible when the vehicle considers itself as a dispatcher (this.isDispatcher()== true) . Choosing this option triggers the allocationMethod().

- *share_context_information*: This option is always feasible unless when the vehicle has to update its schedule. Once this option is chosen, the vehicle broadcasts a message of topic context whose body contains information about its location, current request on-board if any, and the list of known requests.

Allocation Method and Coordination

The allocation method that we implement here is based on calling an ILP solver (e.g. IBM ILOG CPLEX Optimizer[2]). The basic implementation of the allocationMethod() consists in forming the beliefs of the agent into an ILP instance using the solver's Optimization Programming Language. Then running the solver to obtain a solution consisting in a set of allocations for vehicles to requests. The dispatcher agent is then responsible for parsing the solution and transforming it into schedules.

Once the dispatcher agent has the set of schedules, it has to share these schedules with other agents. This is handled by the coordinate() method in which a set of messages are sent to other members. Each message has a specific agent_id as `destination` and the string "schedule" as `topic`. The `payload` of each of these messages contains the new computed schedule that is extracted from the solution for each agent in the connected set.

[2]https://www.ibm.com/analytics/cplex-optimizer

9.2.2 Greedy Decentralized Decisions (Selfish Behavior)

For scenarios in which vehicles behave selfishly, i.e. a vehicle agent does not care about other vehicles' decisions, every vehicle agent has to decide by its own whether it will take a specific request or not based on its own vision about the context of the surrounding environment. We assume for this kind of scenarios that agents can only decide for one request at a time, and they do not have schedules for long term (or in other words, the size of the vehicle schedule is set to one, so it can hold only one request), similar to what van Lon et al. [155] proposed.

Planning Options

The set of planning options for a vehicle agent at a time tick is composed by a set of potential decisions, each of which is dedicated to take one of the known requests as the potential next request for the vehicle. The feasibility of any planning options is determined as follows:

- when a vehicle is carrying a request on board, or when it has chosen a request and is going to serve it, all planning options are infeasible.
- otherwise, the feasibility of deciding on a request r is determined by the possibility of serving the request without violating its constraints. It is a function of the request's time window tw_r and the road temporal distance between the vehicle location and the request origin, in addition to the compliance between the request and the vehicle properties.

Allocation Method

The allocation method here is simple, for each vehicle we have a list of all feasible options, this list is ordered by priority of the concerned request for the vehicle. When this list is not empty, the vehicle agent picks the first request in the list and decides it will be its next request. The priority function is the one described in Chapter 7.

Coordination

In this kind of solution methods, there is no real coordination; the conflicts are resolved upon their appearance by the first arrival policy. Implementation-wise, we consider the function coordinate() simply results in an agreement without sending any message. When a vehicle picks up a request, the request disappears, and if another vehicle was intending to serve the same request, it simply does not receive its information anymore, which make it determine that the request will not be found at its origin, and thus the vehicle becomes empty with no potential request again, so it can choose the new highest priority request.

9.2.3 Cooperative Decentralized Decisions (DCOP)

To implement the cooperative behavior of the DCOP approach, we used FRODO library, which is a Java open-source framework for distributed combinatorial optimization, initially developed at the Artificial Intelligence Laboratory (LIA) of École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. A three-layer, modular architecture is chosen to design FRODO: the algorithms layer builds upon the solution spaces layer and the communication layer in order to provide distributed algorithms to solve DCOPs. FRODO takes in two types of files: files defining optimization problems to be solved, and configuration files defining the nature and the settings of the agents (i.e. the algorithm) to be used to solve them. The benefit of using such a

framework is the separation between problem instances and the algorithms to solve them. The problem instance is described with XCSP 2.1 format [3].

In our experiments, we use FRODO as external library to facilitate the comparison of different solving algorithms without the need of re-implementing them in the agent behavior.

Planning Options

Planning options and their feasibility for DCOP algorithms depend on their implementation characteristics that we illustrated in Chapter 7. However, generally these options concern assigning values to the decision variables at runtime. Here we will not dig in the details of the DCOP algorithms, but we describe the specification of agent behavior to adapt for using FRODO. In our implementation of the DCOPPlanningSubBehavior, we have two special planning options:

- *form_XCSP*: this option is similar to *share_context_information* of the DispatchinPlanningSubBehavior, always feasible unless when the vehicle has to update its schedule. Once this option is chosen, the vehicle broadcasts a message of topic context whose body contains information about its location, current request on-board if any, and the list of known requests. and upon received context each agent build an XCSP representing its local information about the DCOP. Note that we consider here that all vehicles are autonomous and belong to the same fleet, thus sharing the whole DCOP information between CS members is not a subject of privacy issues. However, if privacy is considered, agents should share only the global constraints information, and each agent will have a set of constraints whose information are local to the agent.
- *update_schedule*: this option is only feasible when the solution is computed. It is then the only available option. When chosen the schedule of the vehicle should be replaced by the one received in the message body.

Allocation Method and Coordination

In FRODO, an algorithm is implemented as a policy that describes what should be done upon the reception of such or such message by an agent's queue, and what messages should be sent to other agents, or to another of the agent's modules. This modular design makes algorithms highly and easily customizable, and facilitates code reuse and maintenance. FRODO takes in an agent configuration file that defines the algorithm to be used, and the various settings of the algorithm's parameters when applicable. Thus for each DCOP solving algorithm we have a distinct agent configuration file. As the vehicle properties and the list of known requests form the common knowledge of the agents in the connected set, we assume that the information in the XCSP files for all agents to be identical, and thus any agent can trigger the DCOP solver to build agents schedules as a solution for the local DCOP of the CS. In our experiments the *Lower_ID* agent is the one who triggers it. Once the solution is computed, it is broadcasted to all agents in the CS so that they can replace their schedules by the new ones. If no feasible solution is achieved, no schedule update is performed.

9.2.4 Auction-based Coordinated Decisions (ORNInA)

To experimentally evaluate our contributed *ORNInA* approach, we implement it as a planning sub-behavior of AV agents. The class ORNInAPlanningSubBehavior also extends the abstract PlanningSubBehavior class and implements its abstract methods as follows.

[3] `http://www.cril.univ-artois.fr/~lecoutre/#/benchmarks`

Planning Options

Similar to the SelfishPlanningSubBehavior implementation, we have a set of planning options for a vehicle agent at a time tick composed by a set of potential decisions, each of which is dedicated to make an offer *bid* for inserting one of the known request in the schedule of the vehicle. This stands for the implementation of the auction-based insertion heuristic.

Also, the set of planning options contains *pull_bid* options, standing for the implementation of the *Pull-demand* protocol. A *pull_bid* option is similar to *bid* options, the only difference is that the set of *pull_bid* ones stands for the requests that are already scheduled by other vehicles, while the set of *bid* options is dedicated to unscheduled available known requests.

The feasibility of both kinds of option is dictated by the compliance between the request and both properties and current schedule of the vehicle.

Allocation Method

The allocation method of ORNInA is based on first-price auctions. Similar to the greedy approach, each AV agent has a list of all feasible options, this list is ordered by priority of the concerned request for the vehicle. When this list is not empty, the agent picks the first request in the list and decides to bid for it. The priority function to order the requests is proportional to the marginal cost of inserting the request in the vehicle schedule. Once an AV chooses a request it must coordinate about this decision with other agents.

Coordination

The coordination mechanism is defined by the winner determination mechanism. Once an option is chosen, the method coordinate() is triggered. Depending on if the request is scheduled by another vehicle or not, the AV forms its decision either to broadcast a *bid* or to send a *pull_bid* message. The difference between these two messages is that the *bid* one is broadcasted to the connected set and any agent can make a counter offer to answer it, while the *pull_bid* has a destination agent who is the only one allowed to accept the offer or not. For the *bid* options, an agreement is achieved if no counter offer is received before the offer's timeout. Otherwise, it is a disagreement. For *pull_bid* options, the agreement should be explicitly announced with an *accept_offer* message sent by the agent who hold the request in its schedule.

Summary

In this chapter, we presented *AV-SIM* our experimental framework that implements *AV-OLRA* model and provides the possibility to implement a variety of solution methods. We described the implementation requirements of this framework, and illustrated the abstract and detailed class diagram of how this framework architecture is implemented within *Plateforme Territoire*. We also described the implementation details of the 4 different examples of coordination mechanisms that we compare in this study. In the next chapter, we present and analyze the experimental result of comparing these mechanisms on different types of scenarios.

Chapter 10

Experimental Results

Being autonomous, the vehicles within a fleet can be responsible for their choice of allocation to requests making *decentralised decisions* as we have seen in Chapters 7 and 8, or simply follow the schedules that are *centrally decided by a dispatcher* as shown in Chapter 6. In Chapter 9, we have presented the requirements and guidelines to implement the behaviors of vehicles for the different solution methods using the *AV-SIM* architecture and simulate their performance within *Plateforme Terriroire*. Following these guidelines, we can instantiate the AV-OLRA model with the multiagent model described in Chapter 5, supporting the different types of coordination mechanisms. In this chapter, we present the experimental results of this work. The purpose of these experiments is to assess the feasibility of providing, based on AV-OLRA model, a generic framework for modeling different approaches to the allocation problem in vehicle fleets and then compare their performance on different scenarios.

10.1 Experiments with Synthetic Scenarios

The aim of these experiments was mainly to focus on the feasibility of the *AV-OLRA* model and its scalability for the various solution methods. We were able to create symptomatic situations on which we can tune the different algorithms. We also aimed at achieving an assessment of the performance of our contributed coordination mechanism, namely *ORNInA*, compared to the other ones. In the rest of this section, we present the extraction and post-processing of urban network data, simulation parameters, and the analysis on the obtained results.

10.1.1 Simulation Scenarios

The city map of Saint-Étienne was chosen for the simulation as a medium-scale urban network. We use a unique urban infrastructure map for all our experiments. For a district located between (45.4325,4.3782) and (45.437800,4.387877), more than 1400 edges have been extracted from Open Street Map (OSM), and post-processed by *Plateforme Territoire*. In all of these experiments, we set the number of sources $|S| = 40$, i.e. 40 locations uniformly distributed through the map were selected for being demand emission sources, having a set $E_S \subset E$ of edges, such that $|E_S| = 71$ connecting the sources. Each edge has a number of points which varies according to its length and the information extracted from OSM. The distance between two consecutive points is about 40 meters.

Simulation Parameters

We evaluate the performance of four families of coordination mechanisms: *Selfish*, *Dispatching*, *Market*-based, and *DCOP*. The Java-based MAS and IBM ILOG CPLEX Optimizer Version 12.9.0 have been executed on an octa-core Intel® Core™ i7-8650U CPU @ 1.90GHz, with 32GB DDR4 RAM. For DCOP algorithms, we make use of the implementation from FRODO library [90]. A fleet V of n vehicles is distributed randomly through S at the beginning of execution. Each vehicle $v \in V$ moves (on the points) from one point to the next on the same edge during each simulation cycle. When vehicles have to directly exchange messages, we consider they communicate via DSRC with a realistic communication range of 250 meters. Vehicles are also able to communicate by transitivity to any other vehicle in their connected set. The number of generated requests and the number of vehicles are parameters of the simulation. The passenger requests are generated randomly with pick-up and delivery locations belonging to the set of sources. All scenarios were 1000-cycle long, and at each time cycle, 0 or 1 request is generated. Each instance of these tests is executed 10 times with different random generator seeds.

10.1.2 Selecting and Tuning DCOP Algorithms

In FRODO library, an algorithm is implemented as one or more policies that describe what should be done upon the reception of such or such message by an agent, and what messages should be sent to other agent. FRODO supports a set of algorithms including local search, inference and sampling ones. These algorithms vary also in terms of their optimality.

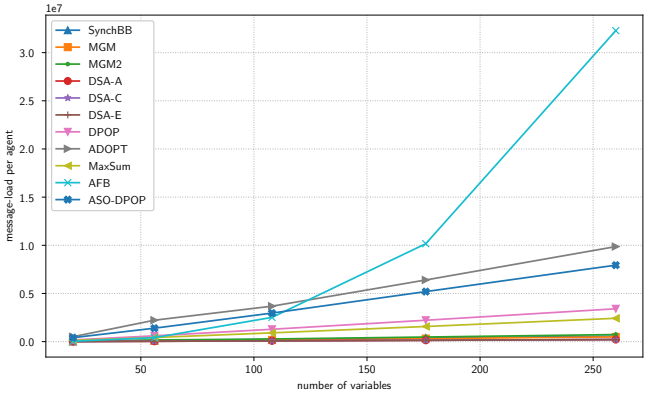
The choice of suitable algorithms and tuning their parameters is not straightforward but depends on the nature of the problem and the computational and communicational constraints. To do so, we have tested with small instances a wide variety of these algorithms, namely:

- Complete:
  - Search: SynchBB, AFB and ADOPT,
  - Inference: DPOP and its variant ASO-DPOP,
- Incomplete
  - Search: DSA (variants A, C, and E), MGM, and MGM-2,
  - Inference: Max-Sum.

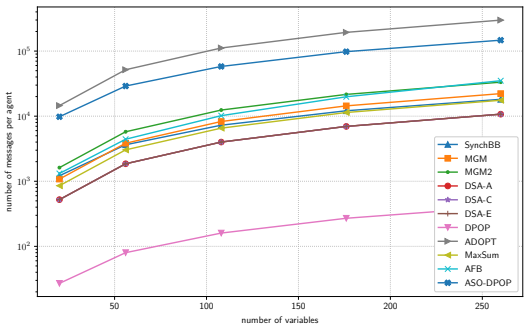
The problem size in these experiments is defined in terms of number of variables nb_{var} that grows with both number of agents and number of requests, and so for the number of constraints. A comparison of the above-mentioned algorithms in terms of communication load and execution time as a function of problem size is shown in Figure 10.1. This comparison allowed to eliminate the ones requiring exponential execution time and/or huge communication load.

ADOPT, DPOP, ASO-DPOP, SynchBB and AFB are known to be complete algorithms that find optimal solutions but have exponential algorithmic complexity. On the other hand Max-Sum is incomplete algorithm with efficient execution time, but it requires an exponential memory size regarding the average number of agent neighbors in the constraint graph, i.e. exponential memory regarding the number of constraints.

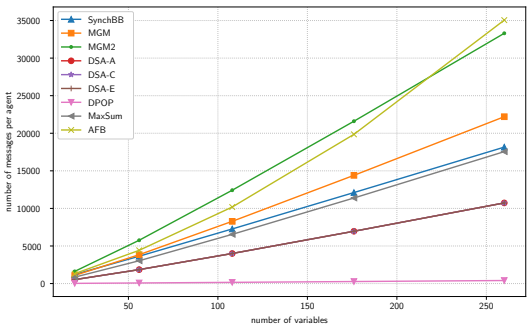
Max-Sum operates on a factor graph that is a bipartite graph in which variables and constraints are represented by nodes. Each node representing a variable in the DCOP is connected to all function nodes that represent the constraints in which it is involved. Similarly, a function node is connected to all variable nodes that represent variables in the original DCOP that are included in the constraint it represents. Each agent adopts the role of the node representing



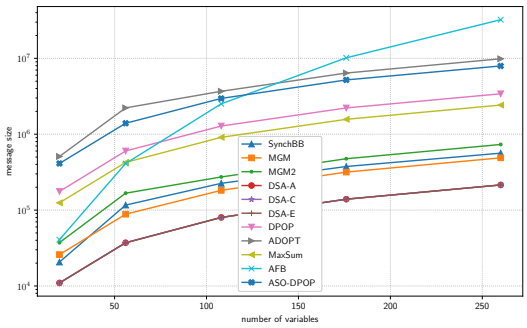
(a) Communication load (all algorithms)



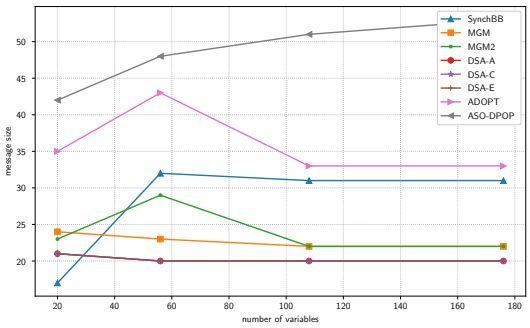
(b) Nb. Msg (all algorithms - log scale)



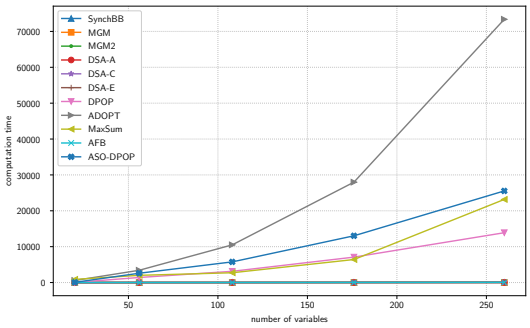
(c) Nb. Msg (ADOPT, ASO-DPOP eliminated)



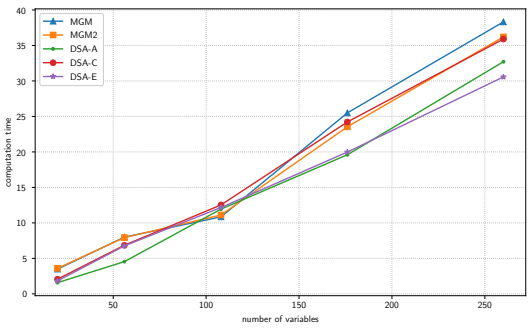
(d) Msg Size (all algorithms - log scale)



(e) Msg Size (AFB, DPOP, MAX-SUM eliminated)

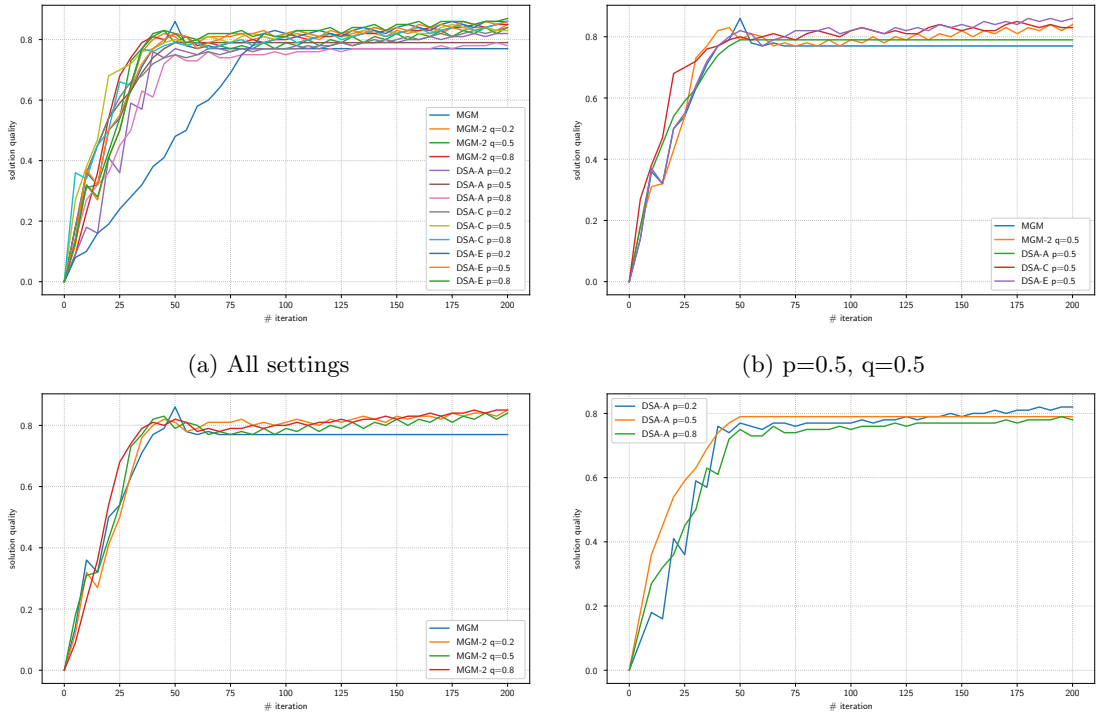


(f) Execution time (all algorithms)



(g) Exec. time (incomplete-search algorithms)

Figure 10.1: Runtime comparison of DCOP algorithms



(a) All settings

(b) p=0.5, q=0.5

(c) MGM

(d) DSA-A

Figure 10.2: Solution quality of local search DCOP algorithms

its own variable and the role of one of the function nodes representing a constraint in which it is involved. Thus, in our case with the highly constrained instances, Max-Sum requires a large amount of memory for the communication load and may not converge because the factor graph will include cycles of different sizes.

The remaining ones are the incomplete local search algorithms whose general structure is synchronous. At each step of the algorithm, an agent sends its assignment to all its neighbors in the constraint network and receives the assignment from all its neighbors. They differ in the method agents use to decide whether to replace their current value assignments to their variables, for example, in the max gain message (MGM) algorithm; the agent that can improve its state the most in its neighborhood replaces its assignment. In MGM-2, the first step is to decide which subset of agents is allowed to make offers. Each agent generates a random number uniformly from $[0, 1]$ and considers itself an offerer if the random number is below a threshold q. If an agent is a offerer, it cannot accept offers from other agents. All agents that are not offerer are considered receivers. Each receiver will randomly (uniformly) choose a neighbor and send it an offer message, which consists of all coordinated moves between the offerer and the receiver that will bring a local utility gain to the receiver in the current context. A stochastic decision to replace an assignment is made by agents in the distributed stochastic algorithm (DSA), similar to MGM-2, each agent in DSA generates a random number from a uniform distribution on $[0, 1]$ and acts if that number is less than some threshold p. The lower threshold value reduces the number of agents who can act at each cycle, which means lower message load and slightly lower solution quality, while higher thresholds means that every agent has more chance to act every iteration, which means more improvement rounds on the solution with their expenses in terms of communication.

A comparison on solution quality of these algorithms is shown in Figure 10.2. This comparison shows that the different variants of DSA perform almost the same on our instances when having the same p value. For both DSA and MGM variants, the quality of solutions grows during the execution time for each cycle until reaching some stabilization point after-which the improvement

becomes very small. By setting $p = 0.5, q = 0.5$, DSA and MGM-2 algorithms reach this stabilization point after around cycles 30 to 50 iterations. More specifically, with variant A of DSA, the stabilization point of 80% is reached on 45 iterations when $p = 0.5$, and MGM-2 reach its 82% on 40 iterations when $q = 0.5$.

Based on these results we limit our experiments on DCOPs to MGM-2 ($q = 0.5$) and DSA (variant A, $p = 0.5$)

10.1.3 Results on Synthetic Data

The results illustrated in this section concern the synthetic scenarios presented in Section 10.1.1. We have executed several instances of problems that vary with the size of the fleet $n \in [4..20]$ over 1000 cycles for each scenario.

Quality of the Solutions

Figures 10.3 and 10.4 illustrate the performance of the five selected coordination mechanisms in terms of QoB and QoS indicators. Every point on these diagrams represents the average, minimum, and maximum indicator value aggregated over 1000 cycles of simulations. They show how the quality of the solution evolves with increasing fleet size.

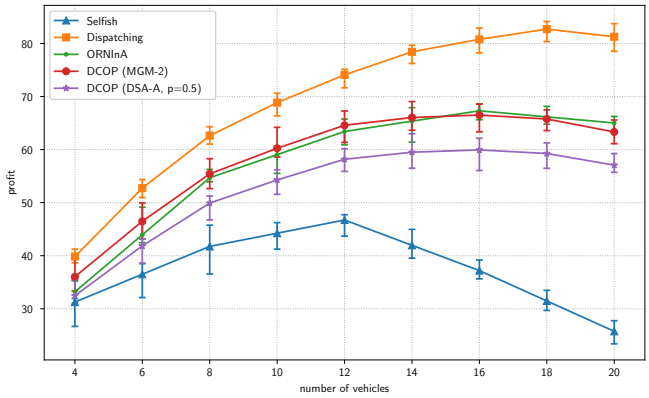


Figure 10.3: QoB evolution with fleet size

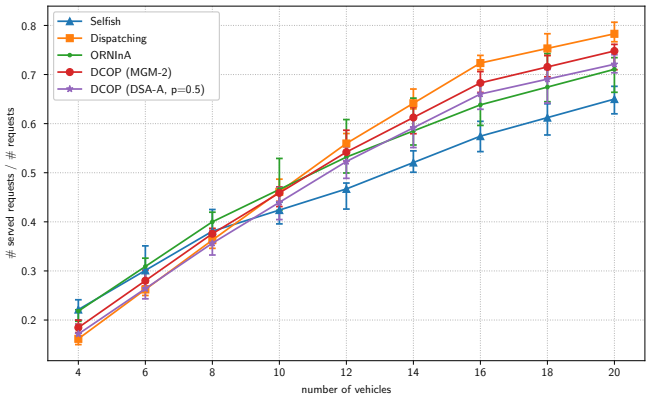


Figure 10.4: QoS evolution with fleet size

We can notice the increase in QoS and QoB with the increasing number of vehicles in the fleet until reaching a threshold of repletion, after which it is not possible to improve the quality by

adding more vehicles. We can still increase the QoS by adding more vehicles, but the amount of increase in QoS achieved by each additional vehicle gradually diminishes, while the operational cost of these vehicles increases, which leads to a decrease in the QoB. The values obtained by the *Dispatching* mechanism represent, to some instinct, an upper-bound for the objective function (QoB) as the central dispatcher calculates for each instance the optimal solution (locally optimal considering the context of the connected set). The performances of the four other mechanisms vary between the indicators. Figures 10.5 and 10.6 show how the quality indicators evolve during the execution of fixed-size fleet scenario. At the beginning vehicles have empty schedules, and any movement means a loss in QoB. Gradually vehicles become aware of more request and add them to their schedules, thus serve them and increase the values of QoB and QoS as a result.

The performance of the *DCOP*s, *ORNInA* and the *Dispatching* mechanisms highly depends on the amount of shared information so that there are no quality discrepancies between the four approaches. With a low number of vehicles, the connected sets are small and, as a consequence, the amount of shared information is reduced. With higher fleet sizes, more information is shared in the connected sets. Additionally, vehicles switch from one CS to another more frequently. The DCOPs and ORNInA approaches perform almost similarly and achieve reasonable values of QoS. To achieve the same values with the *Selfish* behavior, more vehicles in the fleet are required.

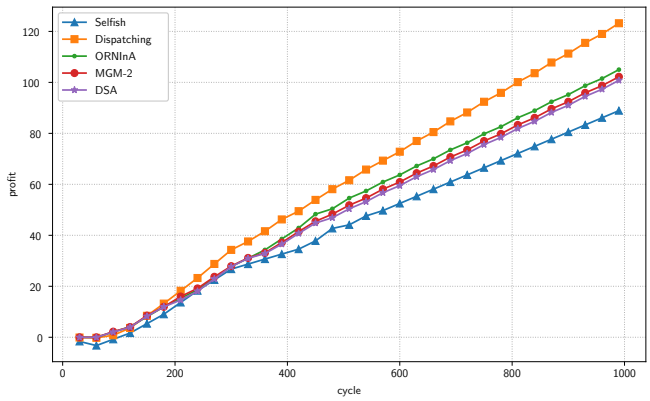


Figure 10.5: QoB evolution during execution

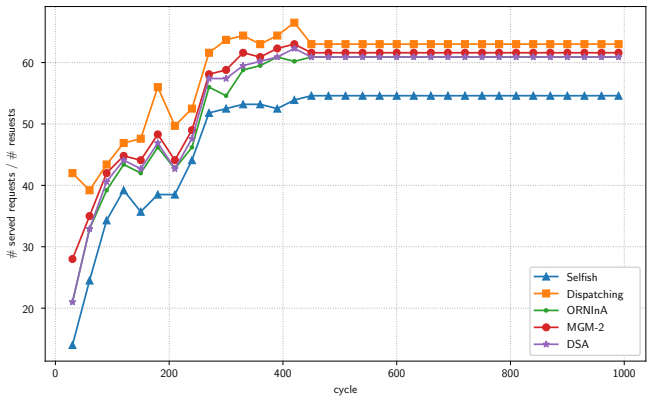


Figure 10.6: QoS evolution during execution

Considering the DCOP algorithms (DSA and MGM-2), at time t, and starting from some feasible solution s_{t-1} achieved at time $t-1$, each of these approaches tries to obtain a new solution s_t in which the value of the objective function is improved. DCOP algorithms focus on maximizing the individual utility of agents to maximize the global objective, which means

assigning to agents requests that increase its gain. The more (potentially successful) trip requests assigned to an agent, the more individual utility is achieved. Of course this will increase QoB, but the main effect will be on the QoS value.

On the other hand, every agent in ORNInA, at each time step, tries to add to its schedule at most one request that can improve its utility and thus QoS and QoB values, then the schedule improvement phase, defined by the pull-demand protocol, tries to reallocate requests in a way that maximizes the global gain even if this reallocation decreases the gain of the agent who abandons the request. The pull-demand protocol affects only the value of QoB as the number of scheduled (to be served) requests does not change, they are only reallocated to other agents. So, while the auction-based coordination of ORNInA performs better than the DCOP search algorithms in QoB, it is outperformed by both of them in terms of QoS.

Communication Load

Table 10.1 shows values of the indicators related to communication obtained by simulating a scenario with 10 vehicles over 1000 cycles with different behaviors. Here the second and third columns report the maximum and average size of exchanged messages (in bytes) representing the **MsgSize** indicator. The fourth column reports the **MsgCount** indicator in terms of the average number of messages received by an agent per simulation cycle.

Even with *Selfish* behavior, agents exchange information messages about the new requests announced. New types of messages are used in the *Dispatching* mechanism: the query and response messages exchanged between the vehicles and the central dispatcher. Query messages are simply the whole context of the connected set of vehicles that ask the dispatcher to build their schedules. Response messages are sent from the dispatcher to the individual vehicles and contain each individual's potential schedule. These messages can be large, depending on the size of the sub-problem.

Bid and answer messages used by the *Auction*-based coordination mechanism are light-weight, so that the values of the **MsgSize** indicator stay close to the no-coordination one, while the **MsgCount** value becomes polynomial in the number of agents in the connected set and number of their known requests.

In the two *Cooperative* coordination mechanisms (DSA and MGM-2), agents in a connected set instantiate a DCOP framework between each other each time they need to decide on a schedule update. Achieving a solution by one of these algorithms requires the exchange of a large number of messages, both of these algorithms are not complete, meaning that they continue their trials to improve the solution until reaching the timeout or local optimum. This will lead to more message exchange. On the other hand, the size of messages exchanged by these two approaches is very small compared to the other approaches.

For each iteration, DCOP agents exchange as many messages as constraints. In average, for our scenarios, each agent sends about 25 messages per iteration. Recalling the findings in Section C.6.2, a stabilization point is achieved after 40 iterations for MGM-2 and 45 iterations for DSA which means at least around 1000 messages are exchanged to achieve such a solution quality. In our experiments, to guarantee this solution quality level, we need to set number of iteration to a value higher than the minimum required, by default in FRODO, this value is (`nbCycles` = 200) thus we kept it, which leads to have 5040 messages per MGM-2 agent, and 5015 messages per DSA agent.

Coordination	max msg size	avg msg size	msg per agent	comm. load	reschedule period
Selfish	140	88	6	2.21 MB	2.0
Dispatching	3500	168	21	11.2 MB	3.0
Auction	140	112	53	37.7 MB	1.5
MGM-2	210	25	5040	297.6 MB	12.0
DSA	236	20	5015	75.1 MB	13.0

Table 10.1: Communication cost and statistics for different coordination mechanisms for scenarios with ten vehicles

Stability of Schedules

Table 10.1 reports the rescheduling period by considering the average interval between two simulation cycles in which vehicles update their schedules. The higher this value is, the more stable the vehicle schedules. In dynamic settings, having stable schedules for a long time means that no new requests are inserted, affecting the QoS. On the other hand, when vehicle schedules change frequently, vehicles may change their destination and oscillate for a while before performing a successful trip, which could decrease QoB. In our scenarios, *DCOP* coordination provides very stable and good quality schedules at the expense of a higher communication load. ORNInA agents, on the other hand, because of their pull-demand protocol, update their schedules more frequently in order to improve the global QoB. If stability is not a constraint, but communication bandwidth is limited, *ORNInA* is still a good candidate.

10.2 Experiments With Real-world Scenarios

The purpose of these experiments is to have more evaluation on the performance of the different approaches and their scalability to real-world instances, and to analyze in-depth the relationship between communication load, stability, completeness, and feasibility of these solutions. To do so, we need to systematically compare performance, quality, feasibility, stability, and technical issues for these approaches in practice.

10.2.1 The New-York City Urban Network

New York City urban network is divided into five boroughs: the Bronx, Brooklyn, Manhattan, Queens and Staten Island in addition to the Newark Liberty International Airport (EWR) neighborhood. New York City's transportation system consists of a complex infrastructure network. Unlike most cities, whose roads are planned in a spoke and wheel layout, New York City's streets and avenues follow a primarily horizontal and vertical cross direction. Manhattan is the most densely populated and geographically the smallest of New York City's five boroughs. It is the urban core of the New York metropolitan area. In Manhattan, there are twelve numbered avenues that run parallel to the Hudson River, and 220 numbered streets that run perpendicular to the river. Similar to the synthetic scenarios in Saint-Étienne, using OpenStreetMap data, we extracted the post-processed Manhattan urban network. This large resulting network, with 4535 arcs, was chosen because cabs and on-demand ride-sharing vehicles are an extremely popular form of transportation in this city, with over 500,000 trips per day.

Starting from 2016, Transport authorities in New York City such as NYC-TLC used geographical zones to encode pickup and delivery locations in their trip records instead of providing the exact geocode of each pickup or delivery location. The NYC network is then composed of 265 zones, from which 69 zones are in Manhattan (see Figure 10.7). As shows the figure, Zones in Manhattan vary in their geographical area; look for example the difference between Central

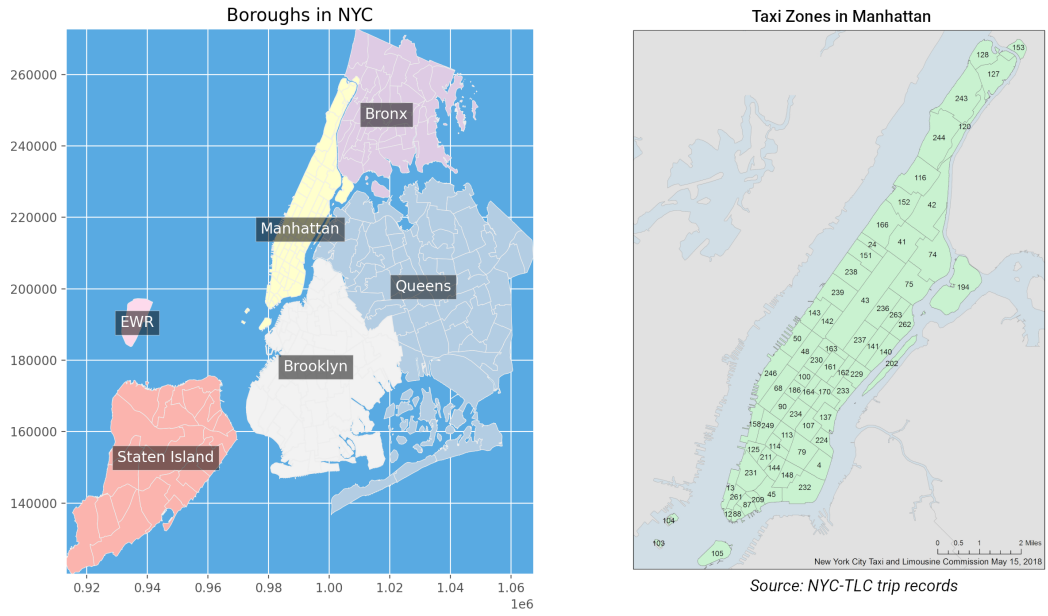


Figure 10.7: Boroughs and Taxi zones in NYC

Park (Zone 43) and its neighboring zone (Zone 163), this means to reach different geographical location in a zone x from a location in a zone y the path lengths could be very different. To cope of with this, we have chosen 100 locations uniformly distributed through these zones for being demand emission sources for their nearby requests.

10.2.2 The NYC-TLC Trip Records Data-Set

The New York City Taxi and Limousine Commission (TLC), established in 1971, is the agency responsible for the licensing and regulation of New York City cabs, for-hire vehicles (community livery, black cars and luxury limousines), commuter vans and paratransit vehicles. TLC collects data, such as trip records, number of vehicles, and fares. Yellow and green taxi trip records include fields recording pick-up and drop-off dates/times, pick-up and drop-off locations, distances traveled, detailed fares, fare types, payment types, and driver-reported passenger counts. The data sets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by licensed technology providers under the Taxicab and Livery Passenger Enhancement Programs.

The data set is provided in form of CSV files per month per vehicle type (Yellow Taxis, Green Taxis, For-Hire Vehicles, and High Volume For-Hire Vehicles). What we are interested in are the traditional Yellow Taxis as they are the most popular and (to the best of our knowledge) have access to operate in any zone in the city. To avoid the particular case of COVID-19 and its impact on the daily operations of small businesses that could affect the taxi demand in the past three years we decided to work on the Yellow Taxi Trip Records from 2018. Each row in the CSV files represents a trip data with 17 capturing: pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. We are interested in only four of them for our experiments:

- `tpep_pickup_datetime`: defines the actual pick-up time,
- `tpep_dropoff_datetime`: defines the actual drop-off time,
- `PULocationID`: defines the pick-up zone id (the origin),
- `DOLocationID`: defines the drop-off zone id (the destination).

Location data including TLC taxi zone location IDs, location names and corresponding boroughs for each ID are provided by NYC-TLC in form of shape files.

10.2.3 Data Analysis on NYC-TLC Trip Records

In NYC there are more than 13k taxis to satisfy the huge density of requests. Thus if we need to achieve a high quality of service in our experiments with any allocation mechanism we need similar fleet size. However, doing microscopic simulation on fleets of such size is not practical. Therefore, we need to produced lower size instances that are well representative for these data. To do so, we studied the statistical model of the data, in what follows we visualize some data analysis on one CSV file from NYC-TLC trip records data set for the month of January 2018. The Python Shapefile Library (pyshp) that provides read and write support for the ESRI Shapefile format, and Matplotlib for 2D plotting library are used for data visualization.

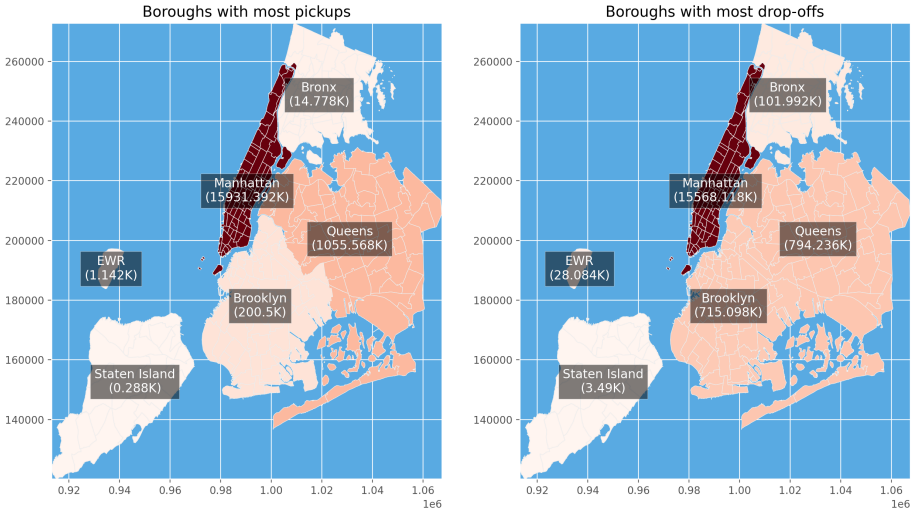


Figure 10.8: Most popular pick-up/drop-off borough in NYC

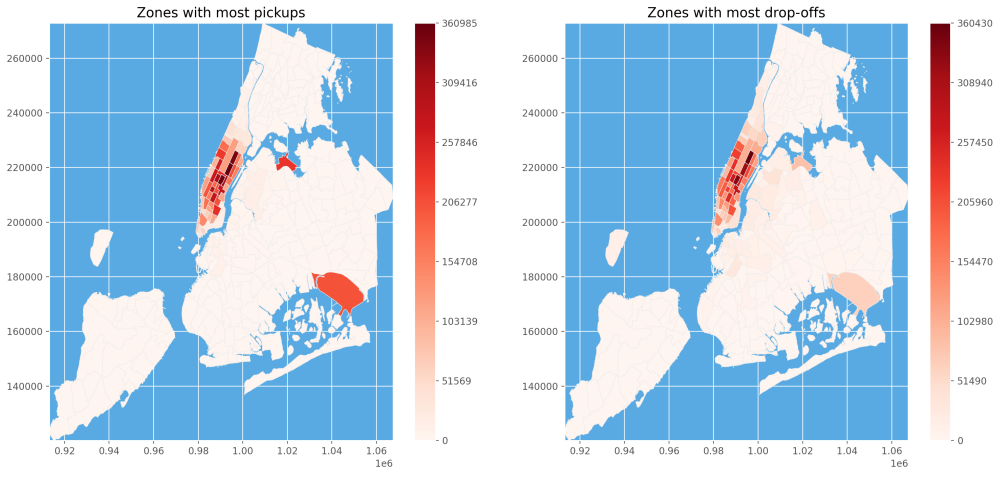


Figure 10.9: Most popular pick-up/drop-off zones in NYC

Spatial distribution of requests

To study the spatial distribution of requests, we aggregate the count of requests over one month by extracting locations of pickup/drop-off and their counts from the CSV file. The results are visualized as heat maps in Figures 10.8 and 10.9. We can see that Manhattan is obviously the most popular borough, containing the top popular pick-up and drop-off zones. Staten Island is the least popular borough. Queens and Brooklyn are also popular, although their pickup/drop-off count is less than 10% of Manhattan's.

Temporal distribution

Here we are interested in identifying the peak hours of pick-up and drop-off, and tracking the number of requests that pops up along the day.

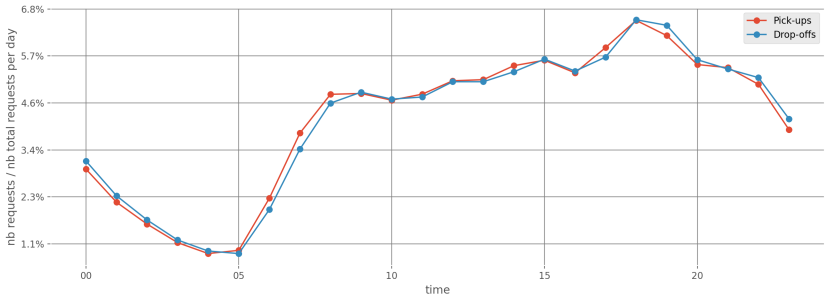


Figure 10.10: The accumulated number of taxi trips at different Time slots

We can see in Figure 10.10, according to the NYC Taxi records of January 2018 that the peak hours are around 6PM to 7PM. And the off-peak hours are around 5AM. The temporal distribution of requests differs also upon the trib distance since the purpose for short trips is not the same as that of long trips.

Based on the findings shown in Figure 10.11, we can observe that for short trips, the peak hours are from 6PM to 10PM. While for long trips (longer than 30 miles), they are from 1PM to 4PM. The off-peak hours are always the same for both cases.

10.2.4 Simulation Parameters

Similar to the synthetic scenarios, we valuated the performance of the same five coordination mechanisms. The Java-based MAS and IBM ILOG CPLEX Optimizer Version 12.9.0 have been executed on a virtual environment using UNIX based calculation server with 12 cores Intel® Xeon® E-2146G CPU @ 3.50GHz and with 32GB DDR4 RAM. We reduced to the half the number of decision cycles of the DCOP algorithms for these experiments in order to reduce the computation time. This was done by setting the attribute `nbCycles` = 100 instead of 200. In all experiments the communication range of vehicles is set to 250 meters.

Based on the statistical model of NYC-TLC trip records shown in the previous section, we produced lower size instances of trip records that holds (almost) a similar spatial and temporal distribution of requests but in lower density. For each instance we chose a limited number of zones, well distributed geographically in Manhattan, and vary in their request density. Then we selected from the Original CSV files a subset of short requests between these zones having pickup/drop-off times corresponding to the temporal distribution of the original data. Figures 10.12a and 10.13 illustrate the spatial and temporal distribution of requests in one of the

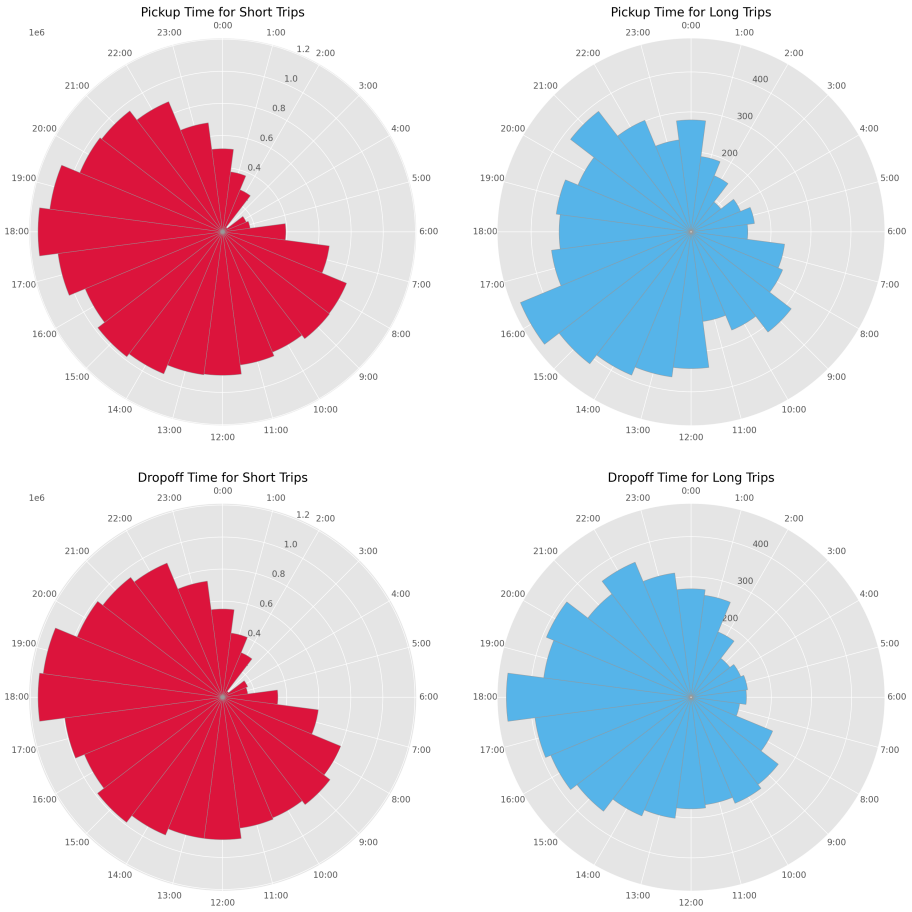


Figure 10.11: Pick-up/Drop-off time for short and long trips

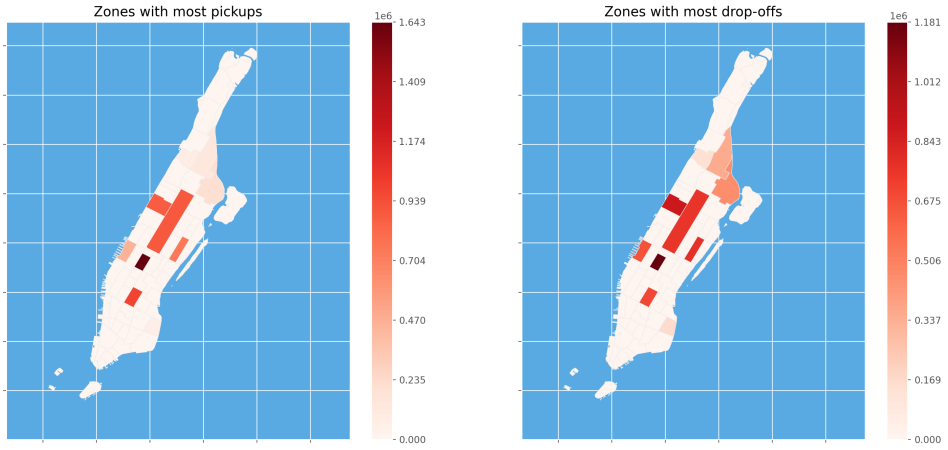
reduced instances. If we consider only Manhattan, the reduced examples are well representative of the original short trip data despite that the number of pick-up and drop-off zones is limited. we have in these examples zones that vary (in the same manner of the original data) from very crowded zones such as Times-square to very uncrowded zones on the boundaries. also we have the same temporal scheme of peak hours.

Passenger requests has been extracted from the CSV files as follows:

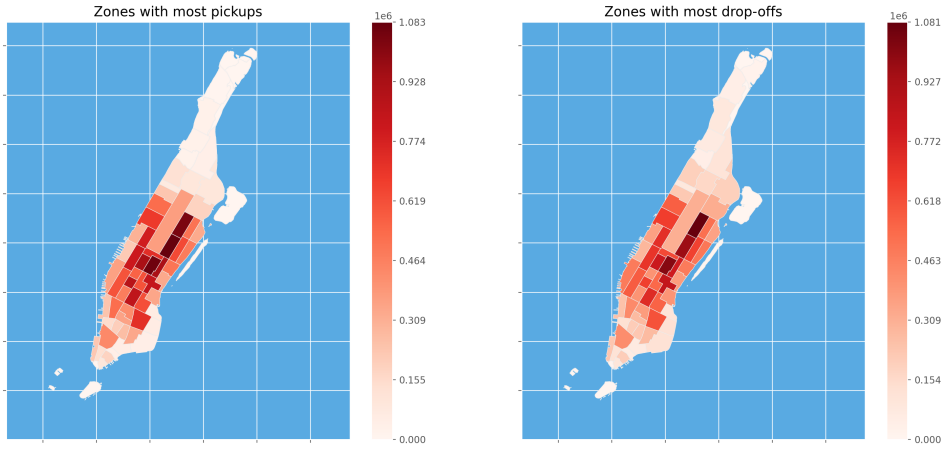
- Having the set of sources S , we define a subset of zone Ids Z of Manhattan network. Each zone must contain at least one source $s \in S$.
- The scenario time interval is defined by date/times of beginning and end :

$$sc_{duration} = [t_{begin}, t_{end}]$$

- The date/time values can be encoded as discrete time ticks using the function `dateToTick(Date date)`
- The length of requests time-windows is a constant twl (by default $twl = 20$ time ticks);



(a) Reduced data set



(b) Original data set

Figure 10.12: Comparing spatial distribution of requests in Manhattan for both datasets

- Using *Pandas*[1] library, a subset of records Rec_{csv} is extracted from the CSV data file; for each record $rec \in Rec_{csv}$ the following conditions should be satisfied:
  - `tpep_pickup_datetime` $\in sc_{duration}$
  - `tpep_dropoff_datetime` $\in sc_{duration}$
  - `PULocationID` $\in Z$
  - `DOLocationID` $\in Z$
- For each record $rex \in Rec_{csv}$, a trip request r is generated having:
  - $o_r = s^{'} \in S$ is a random source located in the zone $z_{PULocationID}$ (having the same id as `PULocationID`)
  - $d_r = s^{''} \in S$ is a random source located in the zone $z_{DOLocationID}$ (having the same id as `DOLocationID`)
  - $tw_r = [t - twl/2, t + twl/2]$ where $t =$ `dateToTick(tpep_pickup_datetime)`

[1] https://pandas.pydata.org/

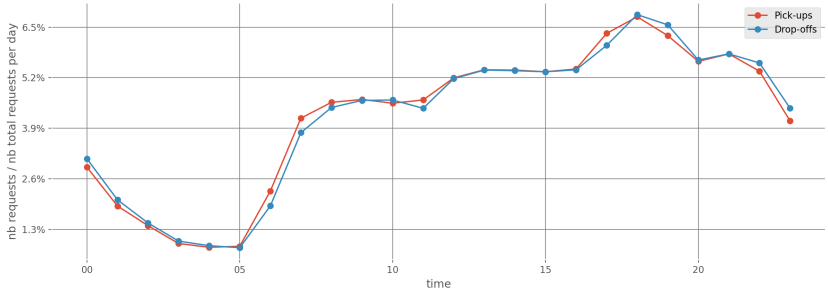


Figure 10.13: temporal distribution of requests in the reduced data set

The `tpep_dropoff_datetime` is not used by the request constructor, It is only used for filtering the CSV records in order to avoid requests that cannot be satisfied during the scenario execution.

The number of vehicles, the set of sources, the input CSV file, and scenario duration are parameters of the experiment. The number of requests during each scenario is determined by the process of request extraction from CSV which is constrained by the set of sources and scenario duration.

10.2.5 Results on NYC Trip Record Scenarios

The results shown in this section are extracted from simulations on real-world scenarios presented in Section 10.2 with larger instances in terms of the network size, fleet size, and number of requests. We also have executed several instances of problems that vary in the size of the fleet $n \in [50..650]$ over 24 hours long (time in data set) each scenario.

Quality of the Solutions

It is not surprising that the evolution of the QoB and QoS for the real-world scenarios shown in Figures 10.14 and 10.15 follows a similar pattern to what is shown in the synthetic data experiments. i.e. the increase in QoS and QoB with the increasing number of vehicles and then reaching a threshold of repletion for QoB. Having a small fleet of m vehicles, for any approach, each vehicle v can contribute in achieving a portion qos_v of the QoS and a portion qob_v of the QoB. qob_v and qos_v refer to the utility of individual vehicles for achieving certain level of solution quality. Increasing the number of vehicles to m' will increase both QoB and QoS values, but the values of portions qob_v and qos_v that a vehicle can contribute to decrease as shown in Figures 10.16 and 10.17. The decrease in qob_v can be explained by the increased operational cost for the new vehicles, while the decrease in qos_v is caused by the inutility of some added vehicles for acheiving the same QoS. This evolution in qob_v and qos_v values continues with growing size of fleet, and thus causes reaching the repletion point.

However, the experiments show that the QoS achieved by fleets notably increases in such scenarios with high density of requests compared to the small instances with lower number of requests. This can be seen more clearly for the *Selfish* fleets and thus supports the hypothesis of van Lon et al. [155] regarding the efficiency of scheduling only one request in advance. The set of options for a vehicle grows linearly with the number of requests in its neighborhood, and thus even if a conflict happened, the vehicle who loses a request can easily switch to next closest one with few extra operational cost.

Anyway, the scheduling approaches based on local search and coordination still providing

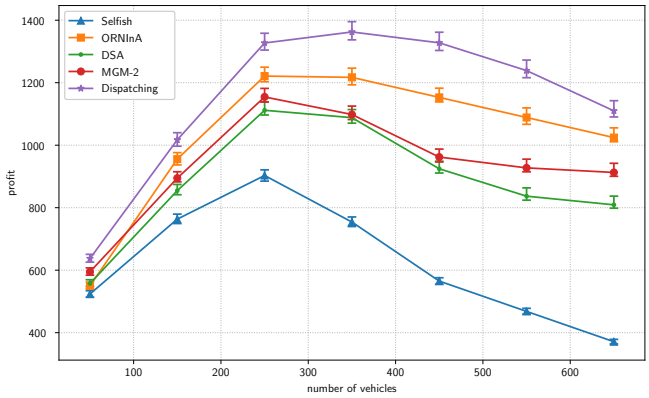


Figure 10.14: QoB vs fleet size for NYC-TLC trips Scenarios

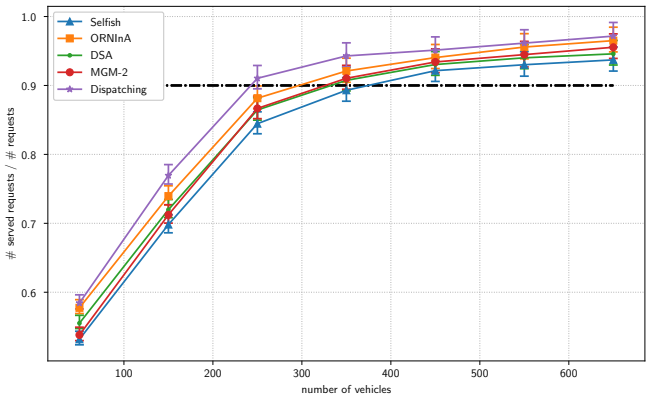


Figure 10.15: QoS vs fleet size for NYC-TLC trips

better quality results in terms of QoS and QoB. We can also notice that even when their QoS levels are so similar, the market-based coordination mechanism (ORNInA) outperforms the DCOP (DSA and MGM-2) algorithms. This does not contradict with the preliminary results we had, as these algorithms keep continuously improving the solution quality until reaching a local optimum. Thus reducing the number of decision can slightly affect the quality of the final solution.

Achieve a certain QoS level requires a minimum number of vehicles, this number varies between the different approaches, for example serving 90% of requests[2] (illustrated by the dashed black line on Figure 10.14) requires optimally 240 vehicles with the *Dispatching* approach, while the number increases to more tan 350 for the *Selfish* one. Figure 10.18 compare the Five approaches in terms of the required fleet size to achieve this 90% QoS. Figure 10.18b shows that the *Selfish* mechanism requires an additional number of vehicles that is almost twice of DCOPs and 3 times of *ORNInA* (regarding the optimal one for Dispatching).

Connectivity

Being distributed through the urban network of Manhattan with surface area about 59km^2 and communication via DSRC with 250m communication range, the fleet is split into a set of connected sets. At the beginning of the execution, the vehicles are distributed randomly around the demand emission sources, then they start moving towards their potential requests. This

[2]In practice serving 90% of requests represent a low QoS. However, this threshold is chosen as an example for comparison purpose, the same comparison can be done with any higher or lower threshold

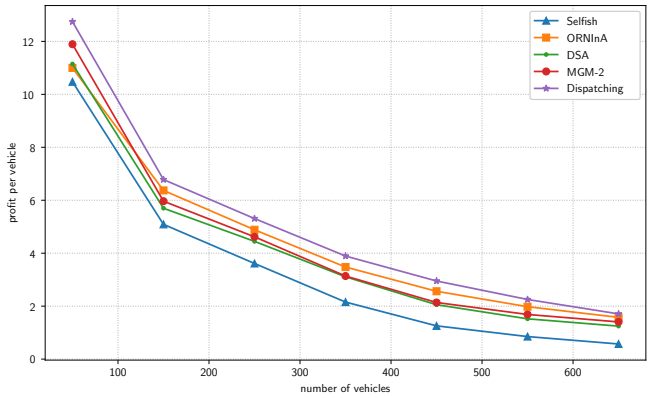


Figure 10.16: Vehicle utility for QoB Evolution for NYC-TLC trips Scenarios

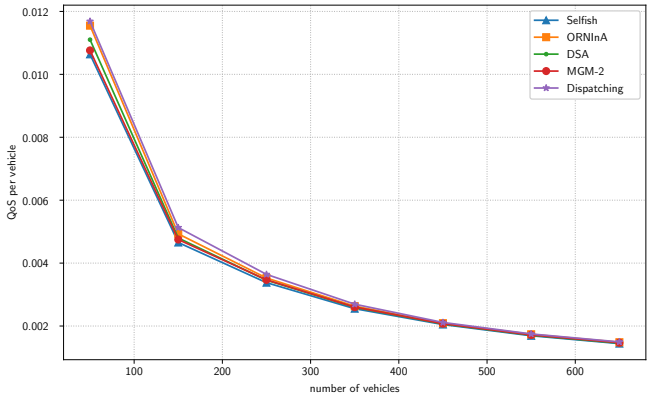
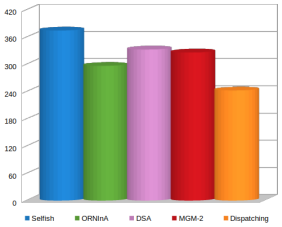


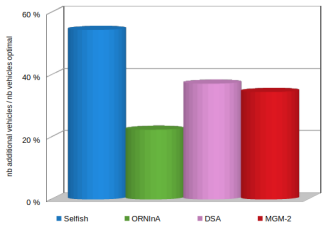
Figure 10.17: Vehicle utility for QoS Evolution for NYC-TLC trips Scenarios

movement affects the structure of the connection graph and make the CSs change dynamically. The number of connected sets is inversely proportional to the connectivity between vehicles, the higher the number of CSs is, the lower the number of vehicles in a single CS.

Figure 10.19 illustrates the evolution of the number of connected sets during the execution of scenarios with 250 vehicles. At the beginning and regarding the random distribution of vehicles, we have about 85 small CSs. When they start to move towards their requests, more vehicles become connected and thus the number of CSs decreases, but still changing in a stable manner. It is worthy to notice that for the *Selfish* fleets, the number of CSs is relatively lower than in other approaches. This means that *Selfish* vehicles keep close to each others. Keeping in mind that Selfish vehicles prefer the closest requests and don't exchange their plans with each others. We can explain this by the fact that vehicles of the same connected set will have the same



(a) nb vehicles



(b) additional vehicles by other approaches to to acheive the same QoS as of Dispatching

Figure 10.18: Required vehicles to serve 90% of requests

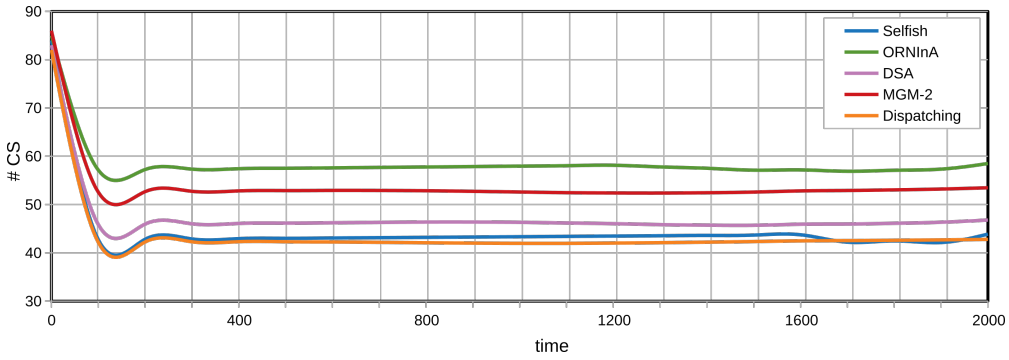


Figure 10.19: Evolution of the number of connected sets during the execution

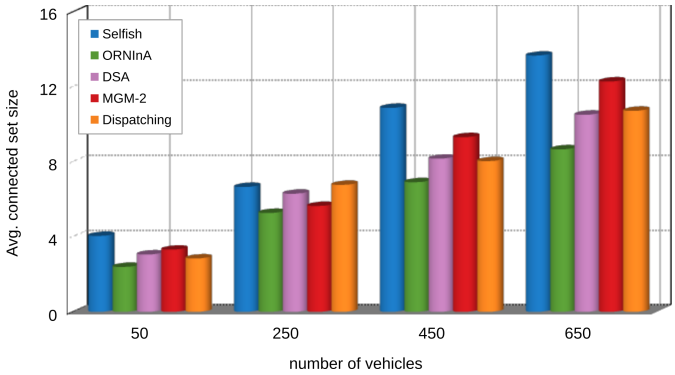


Figure 10.20: Average number of vehicles in connected set

preferences to some extent about the surrounding requests. So there will be for every instance many vehicles going to the same request direction, one of them will manage to pick it up, the rest will have to choose another one and so on. This behavior causes a kind of flocking phenomenon and explains to some extent why the greedy algorithm is less efficient in terms of QoB. On the other hand, the coordination based approaches don't have this behavior. Once a sub-problem is solved each vehicle move in a separate direction to serve its next potential request. Therefore, they may leave and join connected sets more frequently.

When the fleet size grows, the connectivity between vehicles increases. Tracking the evolution of the CS size with varying fleet size, Figure 10.20 illustrates a comparison between the average CS sizes for the five solution methods on scenarios with 50, 250, 450, and 650 vehicles. We can notice that for our experiments the connectivity grows almost linearly with the fleet size for any of these mechanisms.

Communication Load

Figures 10.21 and 10.22 compare the five mechanisms in terms of message load. In Figure 10.21 we illustrate the growing average number of messages (**MsgCount**) received by an agent relative to the number agents in his connected sets. These values represent the **MsgCount** required to solve a single sub-problem instance defined by the CS members and their known requests.

Similar to the preliminary scenarios, the highest **MsgCount** is required by the DCOP algorithms (DSA and MGM-2) while the lower ones are for the Selfish vehicles as their messages only concern the request announcement, and has nothing to do with the vehicle decisions.

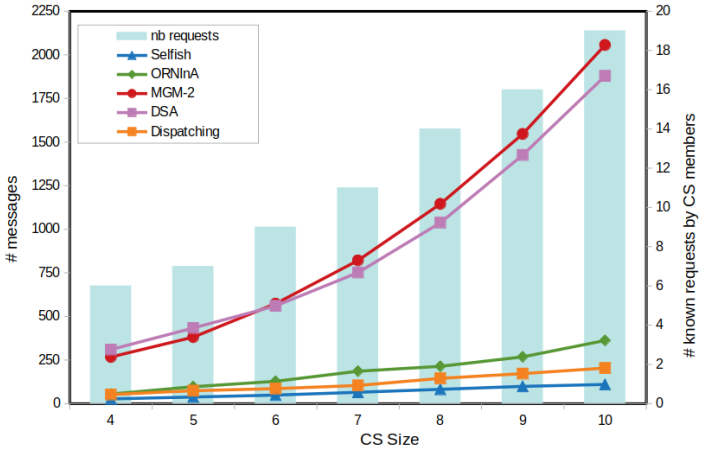


Figure 10.21: Average number of message received by a vehicle in connected set

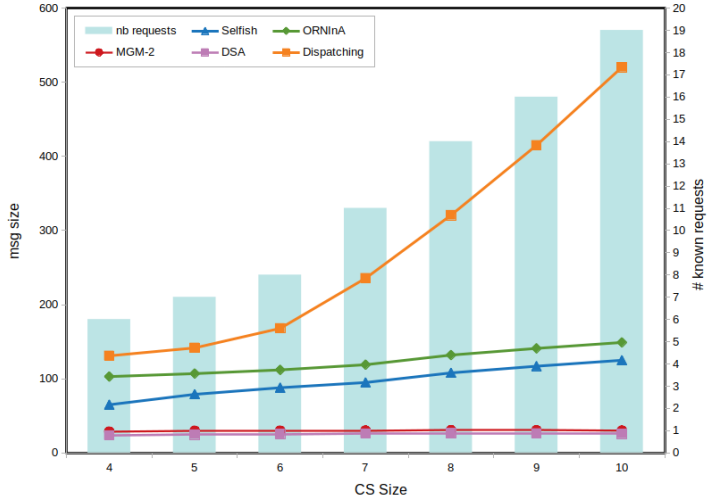


Figure 10.22: Average message size received by a vehicle in connected set

Denoting for time t and for a connected set cs the number of vehicle agents n_{cs}^t and m_{cs}^t the total number of requests known by cs members, **MsgCount** for single sub-problem can be proportional to n_{cs}^t for Selfish, $n_{cs}^t \times m_{cs}^t$ for ORNInA, $2n_{cs}^t$ for Dispatching and ${n_{cs}^t}^2$ for DCOPs. Figure 10.22 illustrate the relative average message size (**MsgSize**). In general, the size of request information messages (which is the common message type for all mechanisms) grows linearly with m_{cs}^t, and thus with the CS size.

For the *Dispatching* mechanism, we have in addition *query* and *response* messages whose size is proportional to $n_{cs}^t \times m_{cs}^t$. Auction and Pull-demand bid messages for ORNInA have stable size that is independent from the sub-problem size, as each of these is one-to-one message concerning only one request at a time. Same for the decisions messages of DCOP, who are small-size. The communication load per simulation cycle of each of the approaches is presented in Table 10.2. The higher size messages for DCOPs are information messages, thus we have the same value of max **MsgSize** for Selfish, MGM-2 and DSA lines. However, the density of DCOPs decision messages highly reduce the effect of information message size on the average **MsgSize** values, so we can see the lines of DSA and MGM-2 in Figure 10.22 as almost constant.

Mechanism	MsgCount	MsgSize	Max **MsgSize** (Bytes)	Rescheduling period
Selfish	$O(n)$	$O(m)$	140	4
ORNInA	$O(n * m)$	$O(200)$	262	3
MGM-2	$O(n^2)$	$O(25)$	140	15
DSA	$O(n^2)$	$O(20)$	140	17
Dispatching	$O(2n)$	$O(n * m)$	30k	5

Table 10.2: Communication load for different approaches

Summary

In this chapter, we aimed at answering **Research Question 4** of this dissertation, which is: *"How can we assess the feasibility and quality of solution methods ?"*
In addition, we aimed at experimentally proving that our contributions all along this dissertation answer to the other research questions.

We illustrated and analyzed the experimental results achieved during this work. First, we have presented the two kinds of scenarios: 1) Synthetic data scenarios in Saint-Étienne urban network, and 2) Real-world data scenarios in New-York City urban network extracted from TLC trip records data set. A data analysis is done on this data set in order to produce lower sized instances that are well representative of the spatio-temporal distribution in this data set. After that we presented the results in form of comparison between five coordination mechanism (Dispatching, Selfish, ORNInA, DCOP with DSA, and DCOP with MGM-2). This comparison has been done in terms of quality of solution metrics (QoS and QoB), Communication load metrics (MsgSize and MsgCount) and Connectivity (CS size and number of CSs).

It is not surprising that the evolution of these metrics follow the same schema for both types of scenarios. However, There was slight differences in some values, this my have been caused by the differences in the density of requests between these scenarios.

In addition to the quantitative and qualitative results obtained by this comparison. These experiment also proved the genericity of the AV-OLRA model (answering to **Research Question 1**) and its multiagent approach that was used to implement the AV-SIM framework and deploy different types of solution methods (answering to **Research Question 2** and **Research Question 3**) then experiment them on different types of scenarios varying in the urban network scale, fleet size, request distribution and request announcement density. These experiments shows that heuristic solutions based on local search and runtime improvement of the quality, like DCOPs (MGM-2 and DSA) or the market-based ORNInA can provide time-efficient, and good quality solutions in dynamic settings (answering to **Research Question 5**).

Conclusion and perspectives

This dissertation presented our work on the resource allocation problem in the on-demand transportation domain. The goal of this work is to explore different methods for solving the problem of allocating autonomous vehicles to online requests in a decentralized manner. Given a fleet of autonomous vehicles deployed in an urban network to meet a large number of passenger requests that arise at runtime in different locations in the city. Without prior knowledge of the spatial or temporal distribution of such demands, the vehicles in the fleet are required to be capable of dynamically updating their schedules to meet newly announced demands. The objective of this work is to model the different aspects of decision-making and optimization problems related to this more general problem. As a result of the modeling of these problems, the question of the choice of centralized and decentralized solution methods arises. In this work, we investigate the directions and compare the performance of distributed constraint optimization (DCOP) techniques, self-organizing multiagent techniques, market-based approaches and centralized operations research solutions.

The problem of allocating resources among multiple entities is a central concern in Computer Science and Economics. In the past few years, allocation problems have become one of the most studied optimization problems in the literature. In Part I, we presented an overview of the available scientific literature on the main aspects related to the problem in question. We provide in Chapter 1 an overview of different vehicle routing problems, specifications of DARP, and their existing solution methods in the operations research literature. In Chapter 2, we review the efforts in the literature of Multiagent Systems and Multiagent Resource Allocation. We conclude from this literature review that MARA solutions are identified by the behaviors of individual agents and their coordination mechanisms so that various solution models exist. We can define three aspects for the characteristics of these solutions which are: 1) the level of autonomy of the agent, 2) the level of cooperation of the agent, and 3) the agent's allocation mechanism. After that, we study the applicability of MARA to ODT in Chapter 3, from which we can infer the need for a unified and adaptive way of representing problem instances. Also, solution methods vary on multiple aspects, so that a unified way of representing and categorizing different solution methods is needed to fairly and effectively compare their performance in different contexts. To the best of our knowledge, such DARP-specific representations have been lacking in the literature of MARA and ODT.

The Contributions of this Dissertation

In this section, we briefly resume the major contributions of this dissertation in answer to the research questions formulated in the Introduction. These contributions have been presented in peer-reviewed, national and international, scientific events (workshops and conferences), and

thus been published as peer-reviewed papers in the proceedings, post-proceedings, or journal special issues of these venues, listed on Page 125. The main hypothesis of this thesis were presented at the doctoral consortium of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21) and included as a short paper it its proceedings [P2].

In Part II, we aimed at filling the aforementioned gap in the literature by proposing AV-OLRA, a generic model for resource allocation problem encountered in the management of autonomous vehicle fleets.

Our first contribution is defining a scalable communication model. Considering the variety of communication technologies that can be used when deploying autonomous vehicle fleets, we needed to define a communication model that scales for every different alternative aiming at answering the **Research Question 3**. We defined this model in Chapter 4 (this contribution is associated with a paper presented in the 12th Workshop on Optimization and Learning in Multiagent Systems [P7], included as a a short paper in the proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems [P8] and included as extended version in the proceedings of the French conference on Multiagent Systems[3] JFSMA-21 [P9]). We started by defining the connectivity between two components in the system, which is achieved by direct messages within limited communication ranges. Being a deployment parameter, the communication range adds another dimension of genericity to AV-OLRA model. To maximize their connectivity, if two vehicles are not close enough to each others to communicate directly, we allow them to communicate transitively upon the existence of another vehicle that is connected directly or transitively to both of them. This led to the definition of connected sets as dynamic sets of entities connected to each other directly or by transitivity.

Our second contribution is to define the main elements requred for the problem formalization. In Chapter 4, we introduced our generic model for online resource allocation problem with autonomous vehicles *AV-OLRA*, aiming to answer **Research Question 1**. This model defines the hypothesis of the problem (components, constraints) and the indicators to evaluate the different allocation strategies (associated publications are [P7], [P8], and [P9]). We formulate the AV-OLRA problem in form of four components: a dynamic set of passenger requests defining the resources; a fleet of autonomous vehicles defining the set of consumers; a directed connected graph representing the urban road network and a time horizon within which vehicles must respond to passenger requests. This model extends a state-of-art model for the Online-Localized Resource allocation (OLRA) adding the specifications of ODT and connected autonomous vehicles. Then, in Chapter 5, we presented the MAS architecture to deliver the AV-OLRA model in which agents can communicate with each other via radio channels using peer-to-peer messages in connected sets. We proposed a multiagent oriented programming requirements build this generic model. Our hypothesis is that the proposed MAS offers genericity on both communication and coordination dimensions.

Our third contribution is proposing a unified representation of solution methods in form of coordination mechanisms. With this contribution presented in [P7], [P8], and [P9] we aimed at answering to **Research Question 2** . AV-OLRA problem can be solved by different methods and algorithms. These methods can be categorized based on the properties of the adopted coordination mechanisms. Thus, they can vary from centralized dispatching to fully decentralized individual decisions, and from cooperative to competitive. In Part III we describe in details the specification of each of these various mechanisms. In Chapter 6, we overview the traditional formulation of the problem as a linear program and describe how to adapt this to the dynamic setting and consider the AV-OLRA sub-problem instances defined by the connected sets. Then in Chapter 7, we explore the direction of decentralization. We have three types of vehicle behaviors here; *Selfish* behavior with no coordination, *Market-based* coordination behavior, and *Cooperative* coordination behavior. We presented the specifications of each of these mechanisms.

[3]Les Journées Francophones sur les Systèmes Multi-Agents (JFSMA)

Our fourth contribution is combining the dynamic responsiveness and solution optimization in one solution method. To provide an answer to the **Research Question 5**, based on the theoretical pros and cons of the presented mechanisms, we proposed in Chapter 8 a new market-based coordination mechanism in which vehicles achieve fast feasible schedules using first price auctions to avoid and handle conflicts. And then starting from these feasible schedules they can initiate peer-to-peer auctions to exchange their scheduled requests in order to improve the quality of solutions. This approach was first presented at the 11th International Workshop on Agents in Traffic and Transportation [P4], and in two French venues (RJCIA-21[4] [P3] and AFIA/ROADEF-21[5] [P1]) organized by the French Association of Artificial Intelligence (AFIA), then been published in the special issue Agents in Traffic and Transportation (ATT 2020) at AI Communication journal [P6].

Our final contribution is proposing a simulation experimental framework built according to AV-OLRA specifications. Part IV was dedicated to experimental validation of the model and to the compare the performance of the solution methods that follow a variety of coordination mechanism. This evaluation was based on the technical and qualitical criteria we defined to asses solution methods aiming to answer **Research Question 4**. In Chapter 9 we overviewed the software engineering behind the experimental framework and the guidelines to implement five coordination mechanisms. Then in Chapter 10 we present the experimental results of the simulation based comparison between the implemented coordination mechanisms. These experiments have been run on two types of scenarios. The first one is based on synthetic data generated on the urban network of Saint-Étienne, while the second is based on real-world data extracted from New York City TLC trip records. The comparison is done in terms of solution quality, communication load, connectivity, and solution stability. Part of these experiments results was included in [P6], [P7], and [P8].

Findings, Limitations, and Future Directions

Simulation results shows that relying on DCOP or auctions to coordinate decentralized decisions provides reasonable quality allocations compared to optimal one-shot allocation and non-coordinated taxis. DCOP-based allocation strategies do not change vehicle schedules too frequently but still induce more communication than the auction-based strategy.

A limitation of our communication model is the phenomena of spatially obscure demands. Those are requests announced far from vehicles and could remain unknown to any connected set for a while until a vehicle passes close to their sources, so they may not be met within their time-window constraints. However, in this work, we considered very dynamic scenarios in the spatial and temporal dimensions so that no such situation would occur in any of our experimental scenarios.

We believe that this work deserves to be further developed; for example, exploring the direction of defining further constraints on vehicle motion to achieve more connectivity between vehicles or to ensure that each emission source is located within the communication space of at least one vehicle.

Another direction to explore is if the vehicles have access to a statistical or machine learning model to predict the future requests, how could that affect their decision quality and what are the effect of such knowledge on the final quality of solution. To improve the quality of AV planing decisions, it could be helpful to use more complex priority functions that consider not only the current context of problem, but also some expectation of how the problem context in the near future will look like. Here arise two questions:

[4]Rencontres des Jeunes Chercheur·ses en Intelligence Artificielle
[5]Le congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision

- Can we make vehicles predict the distribution of requests in space and time?
- How this knowledge could affect the quality of their decisions?

In Appendix A, we introduce a future work proposal to answer these questions.

Finally, we believe that the choice between the various solution methods cannot be considered a straightforward decision. Moreover, these cannot be only seen as technical issues. The need for matching human satisfaction and controllable decisions requires these decisions to be transparent and self-explainable. Our vision on future direction is to apply these findings towards building a fully automatized, self-explainable analytical tool that functions as a recommendation system for resource allocation methods for ODT scenarios. This potential tool takes as input the set of parameters for the scenario (vehicle fleet properties and request distribution model), user's objective function, and preferences, in addition to the environment model (road network and traffic model). Then it recommends specific allocation mechanisms that match the user objectives and preferences. This proposal illustrated in Appendix B has been presented at the 3rd International Workshop on EXplainable and TRAnsparent AI and Multi-Agent Systems (EXTRAAMAS 2021) and included as a book chapter in its post-proceedings [P5].

List of Publications

[P1] A. Daoud. Approche décentralisée d'insertion avec amélioration continue de la qualité de la solution pour un système de transport à la demande. In *5ème journée commune AFIA/ROADEF*, 2020.

[P2] A. Daoud. Multi-agent approach to resource allocation in autonomous vehicle fleets. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4879–4880, 8 2021. Doctoral Consortium.

[P3] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. Approche décentralisée d'insertion avec amélioration continue de la qualité de la solution pour un système TAD. In *Rencontres des Jeunes Chercheur.ses en Intelligence Artificielle (RJCIA)*, Angers, France, July 2020.

[P4] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. Decentralized Insertion Heuristic with Runtime Optimization for On-demand Transport Scheduling. In *11th International Workshop on Agents in Traffic and Transportation*, Santiago de Compostela, Spain, 2020.

[P5] A. Daoud, H. Alqasir, Y. Mualla, A. Najjar, G. Picard, and F. Balbo. Towards explainable recommendations of resource allocation mechanisms in on-demand transport fleets. In *Explainable and Transparent AI and Multi-Agent Systems*, pages 97–115, Cham, 2021. Springer International Publishing. ISBN 978-3-030-82017-6.

[P6] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. ORNInA: A decentralized, auction-based multi-agent coordination in odt systems. *AI Communications*, 34(1):37–53, 2021.

[P7] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. A Generic Agent Model Towards Comparing Resource Allocation Approaches to On-demand Transport with Autonomous Vehicles. In *OptLearnMAS-21: The 12th Workshop on Optimization and Learning in Multiagent Systems*, at AAMAS 2021 (virtual) London, United Kingdom, May 2021.

[P8] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. A Generic Multi-Agent Model for Resource Allocation Strategies in Online On-Demand Transport with Autonomous Vehicles. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, page 3, London, United Kingdom, May 2021.

[P9] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. Un modèle agent générique pour la comparaison d'approches d'allocation de ressources dans le domaine du transport à la demande. In *JFSMA 2021: 29ème Journées Francophones sur les Systèmes Multi-Agents*, pages 127–136, Bordeaux, France, June 2021. Cépaduès.

[P10] Y. Mualla, A. Najjar, A. Daoud, S. Galland, C. Nicolle, A.-U.-H. YASAR, and E. SHAKSHUKI. Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions. *Future Generation Computer Systems*, 100:344–364, 2019.

Bibliography

[1] S. Aalami and L. Kattan. Fair dynamic resource allocation in transit-based evacuation planning. *Transportation research part C: emerging technologies*, 94:307–322, 2018.

[2] 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. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2011.05.017. URL `http://www.sciencedirect.com/science/article/pii/S0191261511000671`. Select Papers from the 19th ISTTT.

[3] AgentPolis. aicenter/agentpolis GitHub, Feb. 2019. URL `https://github.com/aicenter/agentpolis`. online repository: https://github.com/aicenter/agentpolis , accessed 2019-02-20 original repository: https://github.com/agents4its/agentpolis.

[4] S. Airiau and U. Endriss. Multiagent resource allocation with sharable items. *Autonomous Agents and Multi-Agent Systems*, 28:956–985, 2013.

[5] S. Almoustafa, S. Hanafi, and N. Mladenović. New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research*, 226(3):386–394, 2013.

[6] A. Alshamsi, S. Abdallah, and I. Rahwan. Multiagent Self-organization for a Taxi Dispatch System. page 8, May 2009.

[7] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling. Explainable agents and robots: Results from a systematic literature review. In *Proc. of 18th Int. Conf. on Autonomous Agents and MultiAgent Systems*, pages 1078–1088. Int. Foundation for Autonomous Agents and Multiagent Systems, 2019.

[8] K. W. Axhausen and ETH Zürich. *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, Aug. 2016. ISBN 978-1-909188-75-4. doi: 10.5334/baw. URL `http://www.ubiquitypress.com/site/books/10.5334/baw/`.

[9] R. Baldacci, M. Battarra, and D. Vigo. *Routing a Heterogeneous Fleet of Vehicles*, pages 3–27. Springer US, Boston, MA, 2008. ISBN 978-0-387-77778-8. doi: 10.1007/978-0-387-77778-8_1. URL `https://doi.org/10.1007/978-0-387-77778-8_1`.

[10] A. L. Bazzan and F. Klügl. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(3):375–403, 2014.

[11] A. L. C. Bazzan, M. de Brito do Amarante, and F. B. Da Costa. Management of Demand and Routing in Autonomous Personal Transportation. *Journal of Intelligent Transportation Systems*, 16(1):1–11, Jan. 2012. ISSN 1547-2450, 1547-2442. doi: 10.1080/15472450.2012.639635. URL `https://www.tandfonline.com/doi/full/10.1080/15472450.2012.639635`.

[12] O. Beaude, P. Benchimol, S. Gaubert, P. Jacquot, and N. Oudjane. A privacy-preserving method to optimize distributed resource allocation. *SIAM Journal on Optimization*, 30(3): 2303–2336, 2020.

[13] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[14] C. Bellini, G. Dellepiane, and C. Quaglierini. The demand responsive transport services: Italian approach. *Transaction on the Built Environment, WIT Press , www.witpress.com, ISSN 1743-3509*, 64:10, 2003. ISSN 1743-3509. URL `www.witpress.com`.

[15] M. Ben-Akiva, H. Koutsopoulos, and Q. Yang. User's guide for mitsimlab and road network editor (rne). *ITS Program at the Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Boston, USA, Technical Report*, 2002.

[16] C. Bongiovanni, M. Kaspi, and N. Geroliminis. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, 122:436–456, 2019.

[17] E. Borowski and A. Stathopoulos. On-demand ridesourcing for urban emergency evacuation events: An exploration of message content, emotionality, and intersectionality. *International Journal of Disaster Risk Reduction*, 44:101406, Apr. 2020. ISSN 2212-4209. doi: 10.1016/j.ijdrr.2019.101406. URL `https://www.sciencedirect.com/science/article/pii/S221242091930799X`.

[18] G. L. Bradley and B. A. Sparks. Dealing with service failures: The use of explanations. *Journal of Travel & Tourism Marketing*, 26(2):129–143, 2009.

[19] D. Brevet, C. Duhamel, M. Iori, and P. Lacomme. A dial-a-ride problem using private vehicles and alternative nodes. *Journal on Vehicle Routing Algorithms*, 2(2):89–107, 2019.

[20] L. Brunet, H.-L. Choi, and J. How. Consensus-based auction approaches for decentralized task assignment. In *AIAA guidance, navigation and control conference and exhibit*, page 6839, 2008.

[21] D. Calvaresi, Y. Mualla, A. Najjar, S. Galland, and M. Schumacher. Explainable multi-agent systems through blockchain technology. In *Proc. of 1st Int. Workshop on eXplainable TRansparent Autonomous Agents and Multi-Agent Systems (EXTRAAMAS 2019)*, 2019.

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

[23] Canadian Urban Transit Association. Specialized transit eligibility certification programs, overview of canadian and u.s. experience, 2013. URL `http://www.bv.transports.gouv.qc.ca/mono/1136638.pdf`.

[24] D. Cannon and D. A. Martin. Shaping National Disability Policy: Transportation Access and Social Security Reforms, 2004. URL `https://oac.cdlib.org/view?docId=hb2j49n5h3&query=&brand=oac4`.

[25] D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal. A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236(3):833–848, 2014.

[26] D.-N. Chen, B. Jeng, W.-P. Lee, and C.-H. Chuang. An agent-based model for consumer-to-business electronic commerce. *Expert Systems with Applications*, 34(1):469–481, Jan. 2008. ISSN 09574174. doi: 10.1016/j.eswa.2006.09.020. URL `https://linkinghub.elsevier.com/retrieve/pii/S0957417406002983`.

[27] S.-F. Cheng and T. D. Nguyen. TaxiSim: A Multiagent Simulation Platform for Evaluating Taxi Fleet Operations. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 14–21, Lyon, France, Aug. 2011. IEEE. ISBN 978-1-4577-1373-6. doi: 10.1109/WI-IAT.2011.138. URL `http://ieeexplore.ieee.org/document/6040748/`.

[28] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. A. Padget, S. Phelps, J. A. Rodr, and P. Sousa. Issues in Multiagent Resource Allocation. *Informatica*, 30:37, 2006.

[29] G. Cich, L. Knapen, M. Maciejewski, A.-U.-H. Yasar, T. Bellemans, and D. Janssens. Modeling Demand Responsive Transport using SARL and MATSim. *Procedia Computer Science*, 109:1074–1079, 2017. ISSN 18770509. doi: 10.1016/j.procs.2017.05.387. URL `https://linkinghub.elsevier.com/retrieve/pii/S1877050917310566`.

[30] A. Ciortea. *Weaving a Social Web of Things : Enabling Autonomous and Flexible Interaction in the Internet of Things. (Tisser le Web Social des Objets : Permettre une Interaction Autonome et Flexible dans l'Internet des Objets).* PhD thesis, Ecole nationale superieure des mines de Saint-Etienne, France, 2016.

[31] A. Ciortea, O. Boissier, A. Zimmermann, and A. M. Florea. Responsive Decentralized Composition of Service Mashups for the Internet of Things. In *Proceedings of the 6th International Conference on the Internet of Things - IoT'16*, pages 53–61, Stuttgart, Germany, 2016. ACM Press. ISBN 978-1-4503-4814-0. doi: 10.1145/2991561.2991573. URL `http://dl.acm.org/citation.cfm?doid=2991561.2991573`.

[32] N. C. Codella, M. Hind, K. N. Ramamurthy, M. Campbell, A. Dhurandhar, K. R. Varshney, D. Wei, and A. Mojsilovic. Teaching meaningful explanations. *arXiv preprint arXiv:1805.11648*, 2018.

[33] A. Colorni and G. Righini. Modeling and optimizing dynamic dial-a-ride problems. *International transactions in operational research*, 8(2):155–166, 2001.

[34] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, June 2007. ISSN 0254-5330, 1572-9338. URL `http://link.springer.com/10.1007/s10479-007-0170-8`.

[35] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[36] J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M. W. P. Savelsbergh. Chapter 7 Transportation on Demand. In C. Barnhart and G. Laporte, editors, *Handbooks in Operations Research and Management Science*, volume 14 of *Transportation*, pages 429–466. Elsevier, Jan. 2007. URL `http://www.sciencedirect.com/science/article/pii/S0927050706140074`.

[37] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo. Chapter 6 Vehicle Routing. In *Handbooks in Operations Research and Management Science*, volume 14, pages 367–428. Elsevier, 2007. ISBN 978-0-444-51346-5. URL `https://linkinghub.elsevier.com/retrieve/pii/S0927050706140062`.

[38] L. Coslovich, R. Pesenti, and W. Ukovich. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, 175(3):1605–1615, 2006.

[39] P. Cramton, Y. Shoham, and R. Steinberg. An overview of combinatorial auctions. *ACM SIGecom Exchanges*, 7(1):3–14, Dec. 2007. ISSN 15519031. doi: 10.1145/1345037.1345039. URL `http://portal.acm.org/citation.cfm?doid=1345037.1345039`.

[40] 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. ISBN 978-0-9992411-4-1. URL `https://www.ijcai.org/proceedings/2019/31`.

[41] A. Daoud, F. Balbo, P. Gianessi, and G. Picard. Ornina: A decentralized, auction-based multi-agent coordination in odt systems. *AI Communications*, 34(1):37–53, 2021.

[42] F. P. Deflorio, B. D. Chiara, and A. Murro. SIMULATION AND PERFORMANCE OF DRTS IN A REALISTIC ENVIROMENT. page 7, 2002.

[43] C. Desjardins, J. Laumônier, and B. Chaib-draa. Learning Agents for Collaborative Driving. *Multi-Agent Systems for Traffic and Transportation Engineering*, pages 240–260, 2009. doi: 10.4018/978-1-60566-226-8.ch011. URL `https://www.igi-global.com/chapter/learning-agents-collaborative-driving/26941`.

[44] C. di Sciascio, P. Brusilovsky, and E. Veas. A study on user-controllable social exploratory search. In *23rd International conference on intelligent user interfaces*, pages 353–364, 2018.

[45] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006. doi: 10.1109/JPROC.2006.876939.

[46] E. D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. *IFAC Proceedings Volumes*, 20(5):221–226, 1987.

[47] J. Doran, S. Franklin, N. Jennings, and T. Norman. On Cooperation in Multi-Agent Systems. *The Knowledge Engineering Review*, 12, Jan. 2000. doi: 10.1017/S0269888997003111.

[48] C. D. Edwards. *The American Economic Review*, 21(4):701–704, 1931. ISSN 00028282. URL `http://www.jstor.org/stable/504`.

[49] M. Egan and M. Jakob. Market Mechanism Design for Profitable On-Demand Transport Services. *arXiv:1501.01582 [cs]*, Jan. 2015. URL `http://arxiv.org/abs/1501.01582`. arXiv: 1501.01582.

[50] M. Egan and M. Jakob. Market mechanism design for profitable on-demand transport services. *Transportation Research Part B: Methodological*, 89:178–195, 2016.

[51] U. Ehsan, B. Harrison, L. Chan, and M. O. Riedl. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 81–87, 2018.

[52] U. Ehsan, P. Tambwekar, L. Chan, B. Harrison, and M. O. Riedl. Automated rationale generation: a technique for explainable ai and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 263–274, 2019.

[53] 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.

[54] EU_TRAN_Committee. The impact of emerging technologies on the transport system. Technical Report PE 652.226 - November 2020, Policy Department for Structural and Cohesion Policies, Directorate-General for International Policies, European Parliament, 2020.

[55] D. J. Farber and K. C. Larson. The system architecture of the distributed computer system-the communications system. In *Proceedings of the Symposium on Computer Networks (Brooklyn, Apr.). Polytechnic Inst. of Brooklyn, Brooklyn, NY*, 1972.

[56] F. Fioretto, E. Pontelli, and W. Yeoh. Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research*, 61:623–698, Mar. 2018. ISSN 1076-9757. doi: 10.1613/jair.5565. URL `https://jair.org/index.php/jair/article/view/11185`.

[57] T. Fryer. Da vinci brought drawings to life. *Engineering & Technology*, 14(5):18–21, 2019.

[58] V. Furtado, A. Melo, A. Coelho, and R. Menezes. A crime simulation model based on social networks and swarm intelligence. In *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07*, page 56, Seoul, Korea, 2007. ACM Press. ISBN 978-1-59593-480-2. doi: 10.1145/1244002.1244014. URL `http://portal.acm.org/citation.cfm?doid=1244002.1244014`.

[59] B. Gacias and F. Meunier. Design and operation for an electric taxi fleet. *OR Spectrum*, 37 (1):171–194, Jan. 2015. ISSN 0171-6468, 1436-6304. doi: 10.1007/s00291-014-0362-y. URL http://link.springer.com/10.1007/s00291-014-0362-y.

[60] D. Gale and L. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69:9–15, 01 1962. doi: 10.1080/00029890.1962.11989827.

[61] P. Gianessi, A. Ceselli, L. Létocart, and R. W. Calvo. A branch&price&cut algorithm for the vehicle routing problem with intermediate replenishment facilities. *Electronic Notes in Discrete Mathematics*, 55:93–96, 2016.

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

[63] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

[64] J. M. S. Grau and M. A. E. Romeu. Agent Based Modelling for Simulating Taxi Services. *Procedia Computer Science*, 52:902–907, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.05.162. URL https://linkinghub.elsevier.com/retrieve/pii/S187705091500962X.

[65] 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.

[66] T. Gschwind and M. Drexl. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, 53(2):480–491, 2019.

[67] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):93, 2018.

[68] D. Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2(2), 2017.

[69] C. H. Häll, H. Andersson, J. T. Lundgren, and P. Värbrand. The integrated dial-a-ride problem. *Public Transport*, 1(1):39–54, 2009.

[70] R. C. Hampshire and S. Sinha. A simulation study of Peer-to-Peer carsharing. In *2011 IEEE Forum on Integrated and Sustainable Transportation Systems*, pages 159–163, Vienna, Austria, June 2011. IEEE. ISBN 978-1-4577-0990-6. doi: 10.1109/FISTS.2011.5973653. URL http://ieeexplore.ieee.org/document/5973653/.

[71] J. Harri, F. Filali, and C. Bonnet. Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11(4):19–41, 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090403. URL http://ieeexplore.ieee.org/document/5343061/.

[72] J. M. Harris, J. L. Hirst, and M. J. Mossinghoff. *Combinatorics and graph theory*, volume 2. Springer, 2008.

[73] K. Helsgaun. General k-opt submoves for the Lin–Kernighan TSP heuristic. *Mathematical Programming Computation*, 1(2):119–163, Oct. 2009. ISSN 1867-2957.

[74] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

[75] J. Hrnčíř, M. Rovatsos, and M. Jakob. Ridesharing on Timetabled Transport Services: A Multiagent Planning Approach. *Journal of Intelligent Transportation Systems*, 19(1): 89–105, Jan. 2015. ISSN 1547-2450, 1547-2442. doi: 10.1080/15472450.2014.941759. URL https://www.tandfonline.com/doi/full/10.1080/15472450.2014.941759.

[76] L. Hurwicz. The Design of Mechanisms for Resource Allocation. *The American Economic Review*, 63(2):1–30, 1973. ISSN 0002-8282. URL https://www.jstor.org/stable/1817047. Publisher: American Economic Association.

[77] M. J. Osborne. *An Introduction to Game Theory*, volume 3. Oxford university press New York, 2004.

[78] J. J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257, 1986.

[79] N. Jennings, P. Faratin, T. Norman, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *Applied Artificial Intelligence*, 14(2):145–189, 2000. ISSN 0883-9514. doi: 10.1080/088395100117106.

[80] N. R. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated haggling: Building artificial negotiators. In *PRICAI*, page 1, 2000.

[81] X. Jin and L. Jie. A Study Of Multi-Agent Based Model For Urban Intelligent Transport Systems. *International Journal of Advancements in Computing Technology*, 4(6):126–134, Apr. 2012. ISSN 2005-8039, 2233-9337. doi: 10.4156/ijact.vol4.issue6.15. URL http://www.aicit.org/ijact/global/paper_detail.html?jname=IJACT&q=736.

[82] D. Juneja, A. Singh, R. Singh, and S. Mukherjee. A Comprehensive Conflict Resolution Approach for Multiagent Systems. 5(1):12, 2016.

[83] P. Kalina, J. Vokřínek, and V. Mařík. Agents Toward Vehicle Routing Problem With Time Windows. *Journal of Intelligent Transportation Systems*, 19(1):3–17, Jan. 2015. ISSN 1547-2450, 1547-2442. doi: 10.1080/15472450.2014.889953. URL https://www.tandfonline.com/doi/full/10.1080/15472450.2014.889953.

[84] Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. A Collaborative Multiagent Taxi-Dispatch System. *IEEE Transactions on Automation Science and Engineering*, 7(3):607–616, July 2010. ISSN 1545-5955. doi: 10.1109/TASE.2009.2028577. URL http://ieeexplore.ieee.org/document/5286312/.

[85] S. Kraus, A. Azaria, J. Fiosina, M. Greve, N. Hazon, L. Kolbe, T.-B. Lembcke, J. P. Muller, S. Schleibaum, and M. Vollrath. Ai for explaining decisions in multi-agent environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13534–13538, 2020.

[86] V. Krishna. Introduction. In V. Krishna, editor, *Auction Theory*, pages 1 – 10. Academic Press, San Diego, 2003. ISBN 978-0-12-426297-3. doi: https://doi.org/10.1016/B978-012426297-3.50027-8. URL http://www.sciencedirect.com/science/article/pii/B9780124262973500278.

[87] M. Kümmel, F. Busch, and D. Z. Wang. Taxi Dispatching and Stable Marriage. *Procedia Computer Science*, 83:163–170, 2016. ISSN 18770509. doi: 10.1016/j.procs.2016.04.112. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050916301351.

[88] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implementation of a Distributed, Market-based Resource Allocation System. *arXiv:cs/0412038*, Dec. 2004. URL http://arxiv.org/abs/cs/0412038. arXiv: cs/0412038.

[89] A. Lammoglia, R. M. Faye, and D. Josselin. A dynamic cooperation modelling for improving taxi fleet efficiency. In *Multidisciplinary Research on Geographical Information in Europe and Beyond, proceeding of AGILE'2012 International Conference on Geographic Information Science*, AGILE'2012 International Conference on Geographic Information Science, page 6, Avignin, 2012. Jérôme Gensel, Didier Josselin and Danny Vandenbroucke. ISBN 978-90-816960-0-5.

[90] T. Léauté, B. Ottens, and R. Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop (DCR'09)*, pages 160–164, Pasadena, California, USA, July 13 2009. https://frodo-ai.tech.

[91] D.-H. Lee, H. Wang, R. Cheu, and S. Teo. Taxi Dispatch System Based on Current Demands and Real-Time Traffic Conditions. *Transportation Research Record: Journal of the Transportation Research Board*, 1882:193–200, Jan. 2004. ISSN 0361-1981. doi: 10.3141/1882-23. URL http://trrjournalonline.trb.org/doi/10.3141/1882-23.

[92] D. Lehmann, R. Müller, and T. Sandholm. The Winner Determination Problem. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, pages 297–318. The MIT Press, Dec. . ISBN 978-0-262-03342-8. doi: 10.7551/mitpress/9780262033428.003.0013. URL http://mitpress.universitypressscholarship.com/view/10.7551/mitpress/9780262033428.001.0001/upso-9780262033428-chapter-13.

[93] Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[94] J. Liu, H. Jing, and Y. Tang. Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, 136(1):101–144, Mar. 2002. ISSN 00043702. doi: 10.1016/S0004-3702(01)00174-6. URL http://linkinghub.elsevier.com/retrieve/pii/S0004370201001746.

[95] P. Liu, Y.-F. Lin, J.-J. Wu, and Z.-H. Kang. An optimal scheduling algorithm for an agent-based multicast strategy on irregular networks. *The Journal of Supercomputing*, 42 (3):283–302, Oct. 2007. ISSN 0920-8542, 1573-0484. doi: 10.1007/s11227-007-0116-6. URL http://link.springer.com/10.1007/s11227-007-0116-6.

[96] R. Liu, D. V. Vilet, and D. P. Watling. DRACULA: DYNAMIC ROUTE ASSIGNMENT COMBINING USER LEARNING AND MICROSIMULATION. *PTRC*, E:10, 1995.

[97] Y. Liu and Y. Mohamed. Multi-Agent Resource Allocation (MARA) for modeling construction processes. In *2008 Winter Simulation Conference*, pages 2361–2369, Dec. 2008. doi: 10.1109/WSC.2008.4736343.

[98] Y. Liu, Z. Li, J. Liu, and H. Patel. A double standard model for allocating limited emergency medical service vehicle resources ensuring service reliability. *Transportation Research Part C: Emerging Technologies*, 69:120–133, 2016. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2016.05.023. URL https://www.sciencedirect.com/science/article/pii/S0968090X16300602.

[99] J. Ludwig, A. Kalton, and R. Stottler. Explaining complex scheduling decisions. In *IUI Workshops*, 2018.

[100] M. Maciejewski. BENCHMARKING MINIMUM PASSENGER WAITING TIME IN ONLINE TAXI DISPATCHING WITH EXACT OFFLINE OPTIMIZATION METHODS. *Archives of Transport*, 30(2):65–75, June 2014. ISSN 0866-9546, 2300-8830. doi: 10.5604/08669546.1146978. URL https://publisherspanel.com/gicid/01.3001.0003.9666.

[101] M. Maciejewski and J. Bischoff. Large-scale Microscopic Simulation of Taxi Services. *Procedia Computer Science*, 52:358–364, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.05.107. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050915009072.

[102] M. Maciejewski and K. Nagel. The influence of multi-agent cooperation on the efficiency of taxi dispatching. In *International conference on parallel processing and applied mathematics*, pages 751–760. Springer, 2013.

[103] M. Maciejewski and K. Nagel. Simulation and dynamic optimization of taxi services in MATSim. *Transportation Science*, page 34, 2013.

[104] M. Maciejewski, J. Bischoff, and K. Nagel. An Assignment-Based Approach to Efficient Real-Time City-Scale Taxi Dispatching. *IEEE Intelligent Systems*, 31(1):68–77, Jan. 2016. ISSN 1541-1672. doi: 10.1109/MIS.2016.2. URL http://ieeexplore.ieee.org/document/7389909/.

[105] M. Maciejewski, J. Bischoff, S. Hörl, and K. Nagel. Towards a Testbed for Dynamic Vehicle Routing Algorithms. In J. Bajo, Z. Vale, K. Hallenborg, A. P. Rocha, P. Mathieu, P. Pawlewski, E. Del Val, P. Novais, F. Lopes, N. D. Duque Méndez, V. Julián, and J. Holmgren, editors, *Highlights of Practical Applications of Cyber-Physical Multi-Agent Systems*, volume 722, pages 69–79. Springer International Publishing, Cham, 2017. ISBN 978-3-319-60284-4 978-3-319-60285-1. doi: 10.1007/978-3-319-60285-1_6. URL http://link.springer.com/10.1007/978-3-319-60285-1_6.

[106] O. B. Madsen, H. F. Ravn, and J. M. Rygaard. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*, 60(1):193–208, 1995.

[107] S. Manvi and M. Kakkasageri. Multicast routing in mobile ad hoc networks by using a multiagent system. *Information Sciences*, 178(6):1611–1628, 2008. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2007.11.005. URL https://www.sciencedirect.com/science/article/pii/S0020025507005282.

[108] MATSim-simulator. Multi-agent transport simulation, 2007. URL https://matsim.org/. original date: Nov 18, 2007 online repository https://github.com/matsim-org/matsim/.

[109] J. McCarthy. Computer controlled cars. *Obtained from Stanford University http://www-formal.stanford.edu/jmc/progress/cars/cars.html, Computer Science Department, Stanford University, Stanford, CA 94305*, 1968. URL http://www-formal.stanford.edu/jmc/.

[110] M. G. McNally. *The four-step model*. Emerald Group Publishing Limited, 2007.

[111] A. Mingozzi, L. Bianco, and S. Ricciardelli. Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations Research*, 45(3):365–377, 1997.

[112] C. Molnar, G. Casalicchio, and B. Bischl. Interpretable machine learning–a brief history, state-of-the-art and challenges. *arXiv preprint arXiv:2010.09337*, 2020.

[113] Y. Mualla. *Explaining the Behavior of Remote Robots to Humans: An Agent-based Approach*. PhD thesis, University of Burgundy - Franche-Comté, Belfort, France, 2020. URL http://www.theses.fr/2020UBFCA023. 2020UBFCA023.

[114] S. Muelas, A. LaTorre, and J.-M. Peña. A distributed vns algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, 54:110–130, 2015.

[115] A. Muthoo. Sequential bargaining and competition. *Economic Theory*, 3(2):353–363, 1993.

[116] D. Netlogo. NetLogo, 2011. URL https://github.com/NetLogo/NetLogo. Center for Connected Learning, original-date: 2011-10-17 online: https://github.com/NetLogo/NetLogo.

[117] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos. The repast simphony runtime system. In *Proceedings of the agent 2005 conference on generative social processes, models, and mechanisms*, volume 10, pages 13–15. Citeseer, 2005.

[118] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[119] S. N. Parragh, K. F. Doerner, and R. F. Hartl. Demand responsive transportation. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[120] R. V. Parys, M. Verbandt, M. Kotzé, J. Swevers, H. Bruyninckx, J. Philips, and G. Pipeleers. Flexible Multi-Agent System for Distributed Coordination, Transportation & Localisation. page 3, 2018.

[121] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 1446–1451, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[122] 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. doi: 10.3166/RIA.32.223-247.

[123] M. Posada, H. Andersson, and C. H. Häll. The integrated dial-a-ride problem with timetabled fixed route service. *Public Transport*, 9(1-2):217–241, July 2017. ISSN 1866-749X, 1613-7159. URL `http://link.springer.com/10.1007/s12469-016-0128-9`.

[124] A. Preece. Asking 'Why'in AI: Explainability of intelligent systems–perspectives and challenges. *Intelligent Systems in Accounting, Finance and Management*, 25(2):63–72, 2018.

[125] P. Prosser and P. Shaw. Study of greedy search with multiple improvement heuristics for vehicle routing problems. 02 1997.

[126] H. Raiffa. *The art and science of negotiation*. Harvard University Press, 1982.

[127] K. N. Ramamurthy, B. Vinzamuri, Y. Zhang, and A. Dhurandhar. Model agnostic multilevel explanations. *arXiv preprint arXiv:2003.06005*, 2020.

[128] Z. Ren, C. J. Anumba, and O. O. Ugwu. Negotiation in a multi-agent system for construction claims negotiation. *Applied Artificial Intelligence*, 16(5):359–394, May 2002. ISSN 0883-9514, 1087-6545. doi: 10.1080/08839510290030273. URL `http://www.tandfonline.com/doi/abs/10.1080/08839510290030273`.

[129] N. Ronald, R. Thompson, and S. Winter. Simulating demand-responsive transportation: a review of agent-based approaches. *Transport Reviews*, 35(4):404–421, 2015.

[130] A. Rosenfeld and A. Richardson. Explainability in human–agent systems. *Autonomous Agents and Multi-Agent Systems*, pages 1–33, 2019.

[131] J. S. Rosenschein and G. Zlotkin. *Rules of encounter: designing conventions for automated negotiation among computers*. MIT press, 1994.

[132] R. J. F. Rossetti, R. H. Bordini, A. L. C. Bazzan, S. Bampi, R. Liu, and D. V. Vliet. Using BDI agents to improve driver modelling in a commuter scenario. *Transportation Research Part C: Emerging Technologies*, 10(5):373–398, Oct. 2002. ISSN 0968-090X. doi: 10.1016/S0968-090X(02)00027-X. URL `http://www.sciencedirect.com/science/article/pii/S0968090X0200027X`.

[133] S. Rusell and P. Norvig. *Artificial Intelligent: a Modern Approach*. Prentice Hall, 2003.

[134] S. Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

[135] M. Sachenbacher and B. C. Williams. Solving soft constraints by separating optimization and satisfiability. In *Proceedings of the Seventh International Workshop on Preferences and Soft Constraints*, page 15, 2005.

[136] I. SAE. J3016B: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International, 2018. URL `https://www.sae.org/standards/content/j3016_201806/`.

[137] D. Santani, R. K. Balan, and C. J. Woodard. Spatio-temporal efficiency in a taxi dispatch system. In *6th International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2008.

[138] M. W. P. Savelsbergh and M. Sol. The General Pickup and Delivery Problem. *Transportation Science*, 29(1):17–29, Feb. 1995. ISSN 0041-1655, 1526-5447. URL `http://pubsonline.informs.org/doi/abs/10.1287/trsc.29.1.17`.

[139] H. E. Scarf. On the computation of equilibrium prices. 1967.

[140] M. Schilde, K. Doerner, and R. Hartl. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, 38(12): 1719–1730, 2011. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2011.02.006. URL `https://www.sciencedirect.com/science/article/pii/S0305054811000475`.

[141] J. L. Schofer, T. C. R. Program, and N. R. C. U. S. . T. R. Board. *Resource Requirements for Demand-responsive Transportation Services*. Transportation Research Board, 2003. ISBN 978-0-309-08778-0. Google-Books-ID: RG9wnNBKCy4C.

[142] K. T. Seow and D.-H. Lee. Performance of Multiagent Taxi Dispatch on Extended-Runtime Taxi Availability: A Simulation Study. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):231–236, Mar. 2010. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2009.2033128. URL `http://ieeexplore.ieee.org/document/5298955/`.

[143] W. Shen and C. Lopes. Managing Autonomous Mobility on Demand Systems for Better Passenger Experience. *arXiv:1507.02563 [cs]*, 9387:20–35, 2015. URL `http://arxiv.org/abs/1507.02563`. arXiv: 1507.02563.

[144] Y. Shoham. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[145] R. Singh, P. Dourish, P. Howe, T. Miller, L. Sonenberg, E. Velloso, and F. Vetere. Directive explanations for actionable explainability in machine learning applications. *arXiv preprint arXiv:2102.02671*, 2021.

[146] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, 29(12):1104–1113, 1980.

[147] M. M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265, 1987. URL `http://www.jstor.org/stable/170697`.

[148] O. Tatari and M. Skibniewski. INTEGRATED AGENT-BASED CONSTRUCTION EQUIPMENT MANAGEMENT: CONCEPTUAL DESIGN. *Journal of Civil Engineering and Management*, 12(3):231–236, 2006. doi: 10.1080/13923730.2006.9636397. URL `https://www.tandfonline.com/doi/abs/10.1080/13923730.2006.9636397`.

[149] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*, pages 479–510. Springer, 2011.

[150] M. Tlig and N. Bhouri. A multi-agent system for urban traffic and buses regularity control. *Procedia - Social and Behavioral Sciences*, 20:896–905, 2011. ISSN 1877-0428. doi: https://doi.org/10.1016/j.sbspro.2011.08.098. URL `https://www.sciencedirect.com/science/article/pii/S1877042811014789`.

[151] M. Treiber and A. Kesting. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[152] C.-H. Tsai. *Controllability and explainability in a hybrid social recommender system*. PhD thesis, University of Pittsburgh, 2020.

[153] O. Ugwu, M. Kumaraswamy, F. Kung, and S. Ng. Object-oriented framework for durability assessment and life cycle costing of highway bridges. *Automation in Construction*, 14(5): 611–632, Oct. 2005. ISSN 09265805. doi: 10.1016/j.autcon.2005.01.002. URL `https://linkinghub.elsevier.com/retrieve/pii/S0926580505000038`.

[154] S. Vallee, A. Oulamara, and W. Cherif-Khettaf. New online reinsertion approaches for a dynamic dial-a-ride problem. *Journal of Computational Science*, 47, 2020.

[155] 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. ISBN 978-1-4503-1178-6. doi: 10.1145/2330784.2330832. URL `http://dl.acm.org/citation.cfm?doid=2330784.2330832`.

[156] L. Vanneschi and R. Poli. *Genetic Programming — Introduction, Applications, Theory and Open Issues*, pages 709–739. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[157] M. Vinyals, J. A. Rodriguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, May 2011. ISSN 1387-2532, 1573-7454. doi: 10.1007/s10458-010-9132-7. URL `http://link.springer.com/10.1007/s10458-010-9132-7`.

[158] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *J. Artif. Intell. Res.*, 1:1–23, 1993.

[159] S. Winter and S. Nittel. Ad hoc shared-ride trip planning by mobile geosensor networks. *International Journal of Geographical Information Science*, 20(8):899–916, Sept. 2006. ISSN 1365-8816, 1362-3087. doi: 10.1080/13658810600816664. URL `http://www.tandfonline.com/doi/abs/10.1080/13658810600816664`.

[160] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut. A sequence learning model with recurrent neural networks for taxi demand prediction. In *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pages 261–268. IEEE, 2017.

[161] X. Xue, Y. Wang, and Q. Shen. Agent Based Multi-attribute Negotiation for Large-Scale Construction Project Supply Chain Coordination. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops*, pages 531–534, Dec. 2006. doi: 10.1109/WI-IATW.2006.30.

[162] L. Yang, Z. Jieru, C. Jingxin, and T. Zhiyong. Central Decision Intellective Taxi System and Multi Ride Algorithm. In *Proceedings of the 2017 International Conference on Artificial Intelligence, Automation and Control Technologies*, AIACT '17, pages 5:1–5:6, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5231-4. doi: 10.1145/3080845.3080850. URL `http://doi.acm.org/10.1145/3080845.3080850`. event-place: Wuhan, China.

[163] S. Yang, H. Guo, K. Liu, Q. Lin, X. Fang, and Q. Chen. Power market equilibrium analysis based on gradient optimization method. In *2019 IEEE Sustainable Power and Energy Conference (iSPEC)*, pages 2211–2216. IEEE, 2019.

[164] D. Yankov. Discrete Event System Modeling Of Demand Responsive Transportation Systems Operating In Real Time. *Graduate Theses and Dissertations*, Mar. 2008. URL `https://scholarcommons.usf.edu/etd/575`.

[165] W. Yeoh. *Speeding up distributed constraint optimization search algorithms*. Citeseer, 2010.

[166] M. Yokoo. *Distributed constraint satisfaction: foundations of cooperation in multi-agent systems*. Springer Science & Business Media, 2012.

[167] M. Zargayouna, F. Balbo, and K. Ndiaye. Generic model for resource allocation in transportation. application to urban parking management. *Transportation Research Part C: Emerging Technologies*, 71:538 – 554, 2016. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2016.09.002. URL `http://www.sciencedirect.com/science/article/pii/S0968090X16301589`.

[168] K. Zhang, Z. Feng, S. Chen, K. Huang, and G. Wang. A framework for passengers demand prediction and recommendation. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 340–347, 2016. doi: 10.1109/SCC.2016.51.

[169] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1):55 – 87, 2005. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2004.10.004. URL `http://www.sciencedirect.com/science/article/pii/S0004370204001481`. Distributed Constraint Satisfaction.

[170] M. Čertický. Flexible Mobility Services Testbed. GitHub, Dec. 2018. URL `https://github.com/agents4its/mobilitytestbed`. online repository: https://github.com/agents4its/mobilitytestbed , original-date: 2013-08-13.

[171] M. Čertický, M. Jakob, R. Píbil, and Z. Moler. Agent-based Simulation Testbed for On-demand Mobility Services. *Procedia Computer Science*, 32:808–815, 2014. ISSN 18770509. doi: 10.1016/j.procs.2014.05.495. URL `http://linkinghub.elsevier.com/retrieve/pii/S1877050914006954`.

Appendix A

Future Direction: Supporting Allocation Mechanisms with Demand Prediction Models

The selection of a heuristic (priority function) plays a major role in the efficiency of allocation mechanisms especially for greedy algorithms. To improve the quality of AV planing decisions, it could be helpful to use more complex priority functions that consider not only the current context of problem, but also some expectation of how the problem context in the near future will look like. Suppose an agent has the ability to predict a proximate value for the next steps, then the priority of his choices could be affected in order to optimize the quality of solution.

A.1 Illustrative example

Let's consider that a greedy agent wants to solve the problem shown in Figure 7.1. At the beginning he sees only two nodes associated with initial direct gains of 2 for the green and 7 for the blue, he doesn't look for the values of their children nodes. Within the initial settings, the agent chooses 7 and thus his final solution would be the path in blue. Giving the agent the ability to predict an approximation of the children nodes for each option (e.g. predicting the average value of children nodes), he can prioritize his options in different ways. Thus, for each option he calculates the approximation value of children nodes and add it to initial gain, then he choose the best among them. As shows Figure A.1, the average value prediction gives 13.67 to the path on left side and 4 to the one on right side. the potential gains are now 15.67 for choosing the green node and 11 for the blue one. Thus the agent prioritizes the green path and potentially becomes able to reach a global optimum. To apply this kind of approaches to the

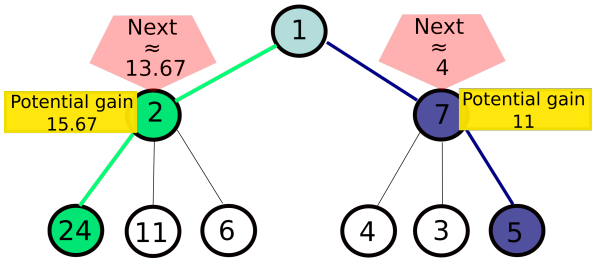


Figure A.1: potential solution improvement when predicting upcomming context (average value prediction)

problem at hand, AV agents should somehow simulate the taxi drivers expertise in predicting the zones that are most likely to have requests during a certain time of the day. Here raise two questions:

1. Can we make vehicles predict, even partially, the distribution of requests in space and time?
2. How this knowledge could affect the quality of their decisions?

To answer the first question, one may consider building a prediction algorithm tat predict the number of potential requests in a certain zone of the city, during a certain time slot in the future. Structuring these predicted values correctly in the agent belief base then could lead to more efficient decisions. Defining the way of structuring this knowledge and affecting the priority function is what answer the second question.

A.2 Request Prediction Models

We aim to build a mechanism to support the agent's decisions by predicting future context information, so that agents can consider to improve the efficiency of their decisions. Obviously no algorithm is capable of making predictions for unlimited time into future. So in terms of time we need to take into consideration that our prediction will be until a certain horizon. Dividing an urban aria into a set of zones Z and sampling the temporal dimension on a sequence of time slots, we can introduce the main hypothesis of the potential model as:

"Given at a time slot t the sequence $n^z_{t-p}, ..., n^z_t$ of passenger numbers during the last p time slots in a certain zone $z \in Z$, can we predict n' the approximate number of passengers in the same zone z at next time slot $t+1$?"

Designing an efficient prediction model and proving its accuracy and performance is out of the scope of this dissertation. However we can find in the literature of Machine Learning and Data Science several applicable prediction models that can fit with our needs. An interesting contribution here is the sequence learning model proposed by Xu et al. This model is based on recurrent neural networks (RNN)[1] for learning the pattern of taxi demand occurrence in the future based on the requests in the past. Since a model that can learn time series data is necessary here. To let the model learn time series data they used Long short-term memory (LSTM) Layers. The model was capable to continuously make real-time prediction of taxi demand for the entire New-Yourk city with a high level of accuracy [160]. Some other approaches are based on data analysis, for instance Zhang et al. propose a passenger hot-spots recommendation system by analyzing the existing taxi trips. The hot-spots represent the most likely zones to be crowded with passenger requests. They extract hot-spots in each time step and assign a hotness score for each one. This score will be predicted in each time step and combined with the taxi's location. Then the top k hot-spots are recommended [168].

A.3 Planning Options utilities

For our problem scenarios, the vehicle decisions can be supported by information gathered from an efficient request prediction model, such as the sequential learning proposed by Xu et al.. Having an access to such a model allows vehicle to update its belief base at every time tick with an approximate number of potential future requests for any geographical zone of interest. These beliefs are then used as additional factors for computing evaluating the utility of vehicle

[1] RNN is a special type of neural network designed for sequence problems

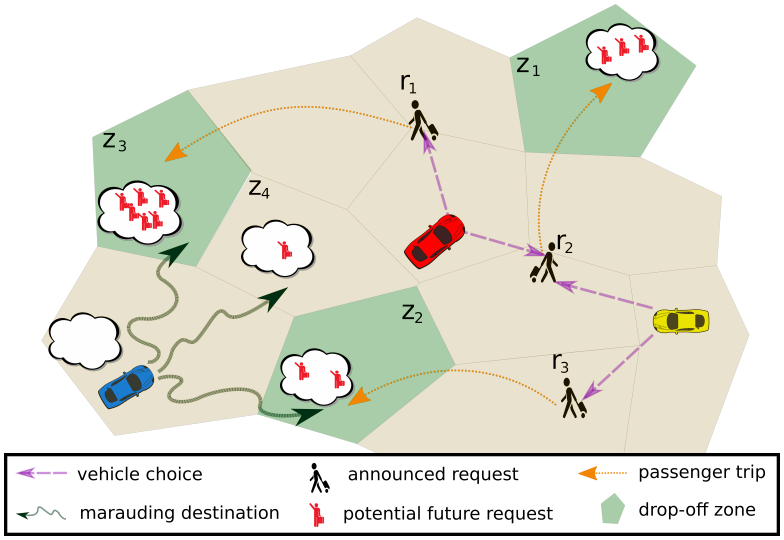


Figure A.2: Vehicle planning options when future requests are predictable

decisions. In what follows we consider vehicles with *Selfish* behavior to explain how can request prediction affect the planning option utilities. Figure A.2 illustrates a snapshot of planning options for three vehicles having access to such a prediction model. This example fits with any decentralized solution method in which vehicles prioritize their options based on their utilities. However, for the seek of simplicity here, we consider the three AVs to have the selfish behavior explained Section 7.1.2. We can distinguish two cases:

- **The vehicle is marauding** when it is not aware of any announced request, in the initial settings it moves randomly towards other zone. However, by predicting the occurrence of potential requests in the surrounding zones, it can prioritize the zone the most likely to have requests to be its marauding destination. In this example the blue vehicle will prioritize z_3 among others.
- **The vehicle is aware of announced requests** thus it calculates the priority of requests not only based on the distance from the pickup location, but also using predicted information about the drop-off zone. An agent assigns a score sc_z^t for each zone, representing the normalized value of its potential crowdedness between 0 and 1. The priority function includes as always a value proportional to the cost, but it also consider the additional information. It becomes a weighted sum of these values.

$$priority_v^t(r) = \frac{a}{cost_v^t(r)} + b.sc_z^t \mid d_r \in z$$

where a and b are parametric weighting factors that defines the importance of operational cost and crowdedness score in computing the priority ($a + b = 1$).

Having equal cost choices to serve r_1 or r_2, the red vehicle in Figure A.2 may prioritize r_1 as its drop-off location d_{r_1} belongs to z_3 in which the probability to have future request is higher than z_1 that contains d_{r_2}. In the same way the yellow vehicle prioritize r_2 over r_3.

Appendix B

Future Direction: A Proposal Towards Explainable Recommender System for ODT Coordination Mechanisms

We aim to design an analytical tool that functions as a recommendation system for on-demand transport (ODT) authorities. This tool recommends specific allocation mechanisms that match the authority's objectives and preferences to solve allocation problems for particular contextual scenarios. The proposal emphasizes the need for transparency and explainability of resource allocation decisions in ODT systems to be understandable by humans and move toward a more controllable resource allocation. We propose in this preliminary work a multiagent architecture and general implementation guidelines towards meeting these requirements.

There are several stakeholders involved in ODT systems including passengers, drivers, service providers, etc.). What we mean by the term *User* in this document is a human user representing the transport authority and looking for the best solution method to solve the problem regarding the authority's preferences and the actual context parameters.

This potential tool takes as input the set of parameters for the scenario (vehicle fleet properties and request distribution model), user's objective function, and preferences, in addition to the environment model (road network and traffic model).
This system simulates the problem scenario and its solutions with different classes of AI methods, then produces to the user the recommended solution model (the solution method and its tuned parameters) that produce results matching the user objective and preferences for the input scenario.

In future Artificial Intelligence (AI) systems, it is vital to guarantee a smooth human-agent interaction, as it is not straightforward for humans to understand the agent's state of mind, and explainability is an indispensable ingredient for such interaction [113]. Recent works in the literature highlighted explainability as one of the cornerstones for building trustworthily responsible and acceptable AI systems [93, 124, 130]. Consequently, the emerging research field of eXplainable Artificial Intelligence (XAI) gained momentum both in academia and industry [67, 7, 21]. XAI is allowing, through explanations, users to understand, trust, and effectively manage the next generation of AI solutions [68].

Providing users with some form of control over the recommendation process can be realized by allowing them to tell the system what they like or by engaging them in adjusting the recommendation profile to synthesize recommendations from different sources [152]. High-quality

explanations allow a better understanding of the results and help the user to make the right decisions. Reliable answers increase confidence in the system, while explanations that reflect system inaccuracies allow the user to modify the system's reasoning or control the weighting parameter that reorganizes or regenerates recommendations.

B.1 About the Need for Explainability

The human perspective is what differentiates ODT from most routing and transport problems; in addition to the technical factors, the quality of the service is influenced by human satisfaction factors, including the stability of service quality, service availability, wait-time, information privacy, passengers' special constraints, and preferences [35].
The following examples show that global system decisions may not fit all stakeholders' preferences: a decision may make some people dissatisfied.

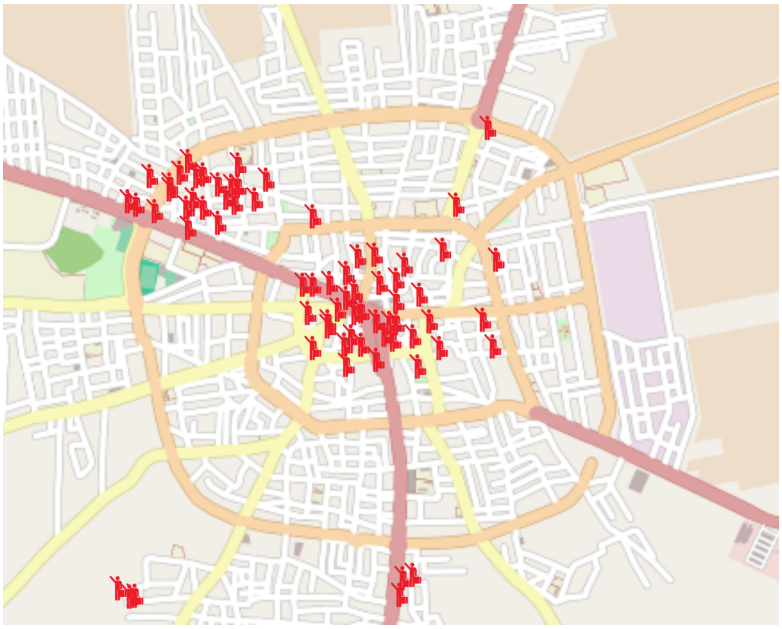


Figure B.1: Passenger request distribution at rush hours.

Scenario 1: Dial-a-ride in rush hours. At rush hours, taxi-ride demand is usually concentrated at specific parts of the city, e.g., city center and train stations, as seen in Fig. B.1. The objective of the transport authority is to maximize the number of satisfied requests while reducing operational costs. An efficient allocation mechanism will dispatch as many vehicles as possible to the crowded areas to serve passengers, prioritizing the requests whose destinations are near other crowded areas. As a consequence, in this example, most of the vehicles move back and forth between the two areas, which reduces the chance of far passengers and makes them wait for a long time for being served, regardless of the urgency level of their requests that may be higher than those who do their ordinary work-home trips from the city center.

Scenario 2: Emergency management ODT. The example of Fig. B.2, introduced by Aalami and Kattan [1] represents a disaster management situation. However, this kind of emergency transport can be modeled as an ODT system [141, 164, 17]. In this example, a failure in facility X leads to a leak of toxic substances. The leakage grows over time and threatens both communities A and B. The inhabitants of these communities need to be relocated to refuge R as soon as possible. A fleet of shuttles is available to relocate people. However, suppose that the

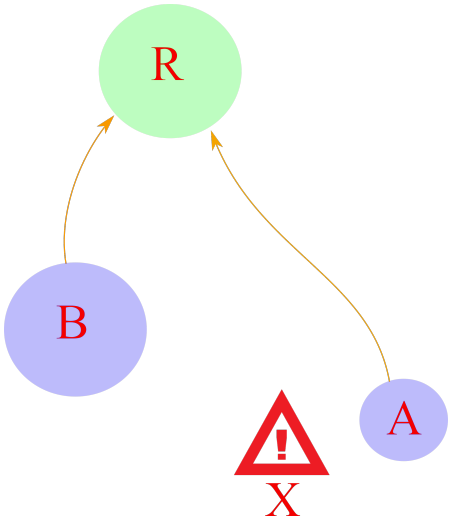


Figure B.2: An example of an emergency scenario.

fleet's size is not large enough to evacuate either community in one hour. Because of the wind direction, the time it takes for the substance to reach community A is double that of community B; however, community A's toxic density will be higher than in community B (assuming the density degrades with distance). Also, community B's population is three times the population of community A. The round-trip time from community A to the refuge is twice the round-trip time from community B to the refuge. In other words, a shuttle assigned to community B can carry twice the number of evacuees compared to the same shuttle assigned to community A. If the goal is to maximize the number of evacuees moved to the refuge within one hour, the answer would be to assign the entire fleet to community B since the round trip time is shorter for this community. However, if the goal is to evacuate high-risk individuals as quickly as possible, the answer would be to assign the entire fleet to community A. While both of these responses seem correct for the corresponding objective, neither seems fair.

Providing explanations for the system decision may increase people's satisfaction [18], and maintain the AI system's acceptability. When a recommendation mechanism is too complicated for lay users, the system may need to justify why the recommendation has been made [44, 149]. The EU General Data Protection Regulation introduces a right of explanation for citizens to obtain "meaningful information about the logic involved" for automated decisions [63]. Generating explanations of autonomous decisions in multiagent environments is even more difficult than providing explanations in other contexts [85]. In addition to identifying the technical reasons that led to the decision, it is necessary to convey the agents' preferences. It is necessary to decide what to reveal about other agents' preferences to increase user satisfaction while considering other agents' privacy, and how those features led to the final decision.

To provide useful explanations, it is necessary to identify the features of the context and decisions relevant to a specific user. Given these features, other relevant agents' preferences should be identified, and any relevant statements that touch on important concepts such as fairness should be generated. Using these features, preferences, and concepts, various explanations could be generated using subsets of them. The selected subset should be transferred in a certain communication form. The personalization of explanations could also be used at this stage since explanations are subjective and depend on multiple factors [145]. As to personalize explanations, there is a need to build a user, or mental model [74] that influences the generation of explanations.

In our resource allocation scenario in vehicle fleets, the allocation process can provide a set of constraints that lead to the proposed allocation. It will be necessary to identify the relevant constraints and generalize statements related to other agents' preferences and general system

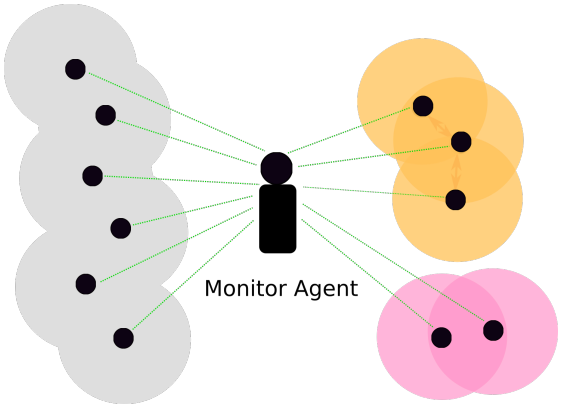


Figure B.3: *MA* and *AV* agents interaction.

constraints related to fairness [99]. Then, we can use user satisfaction models to choose the best constraints, and generalized statements to present.

B.2 Explainable MAS for AV-OLRA recommendation

In this section, we introduce *EX-AV-OLRA*, an extension to *AV-OLRA* metamodel with explainability-related components. We present a multiagent model for an explainable recommendation system that realizes the *EX-AV-OLRA* model. we propose to formulate the *EX-AV-OLRA* model as:

$$\textit{EX-AV-OLRA} := \left(\mathcal{R}, \mathcal{V}, \mathcal{G}, \mathcal{T}, \mathcal{X}\right) \tag{B.1}$$

Where $\left(\mathcal{R}, \mathcal{V}, \mathcal{G}, \mathcal{T}\right)$ define an *AV-OLRA* and $\mathcal{X}$ defines the explaining mechanism.

We aim to design a recommender system in which a human user sets the scenario parameters to create an AV-OLRA instance, setting objective and utility preferences. The system's output is a recommendation to use the solution method that is the best match to user preferences, supported with multi-level explanations of why particular methods are recommended and why others are discouraged.

The multiagent model for the explainable recommender system extends the AV-OLRA model. An additional agent type *Monitor Agent (MA)* plays the role of proxy for *AV*s to produce human-readable personalized explanations for the recommended methods. Unlike the inter-*AV*s' limited-range communication model, the *MA* can interact with *AV*s globally (See Fig. B.3).

This interaction is only to monitor the performance of *AV* agents and logging the explanations of their actions during the simulation. The *MA*'s role is to aggregate the explanations of *AV*s' actions during the simulation to enable building explainable recommendations. In practice, only the *AV*s are deployed. Their communication constraints in real world scenarios should be taken into account during the simulation. To do so, this global interaction means should never be used for communication between *AV*s.

In this work, we propose to add another sub-behavior (the *explaining* sub-behavior) to the *AV* model. This sub-behavior consists of two phases *generating explanation* and *monitor-agent interacting* as shown in Fig. B.4.

Generating explanation phase is triggered whenever a decision is taken (in *planning* sub-behavior). The *AV* gathers all information related to the taken decision (the leading constraints, context information, potential improvement in the solution quality, etc.), in addition to the changed decision variables and their values. This information, together with the contextual data gathered in the previous steps from the agent belief base, are used to generate an understandable

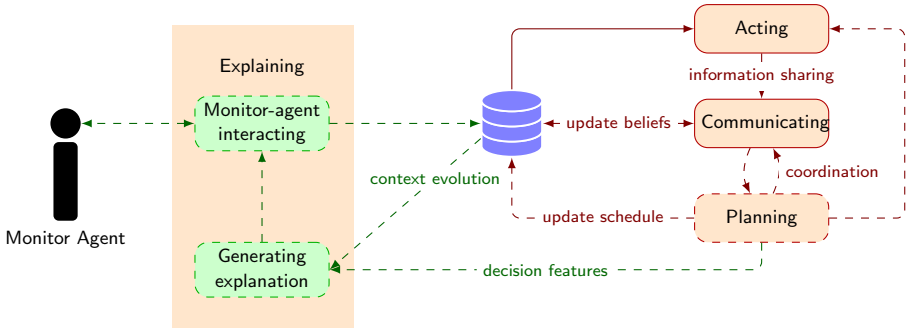


Figure B.4: Explainable AV agent behavior in *EX-AV-OLRA*

Solution	*DA*	*AC*	*AM*	**Explanation examples**
Selfish	*D*	*N*	Greedy	Why prioritizing a specific request?
Dispatching	*C*	*S*	MILP	Which constraints are violated?
Market	*D*	*S*	Auctions	How winner determination computed?
				Why accepting some trade options?
Cooperative	*D*	*S*	DCOP	What are individual costs and utilities?

Table B.1: Examples of solution models and what should be explained.

explanation for the taken decision. When the explanation is generated, the agent moves to the *monitor-agent interacting* phase. In the *monitor-agent interacting* phase, the generated explanations are sent to the monitor agent and stored in the *AV* belief base. To reflect the behavior of *AV*s in real world scenarios, the *MA* should never play the role of communication mediator between *AV*s.

The set of possibly explainable actions and decisions depend basically on the chosen solution model. Table B.1. lists some examples of solution models in line with their possibly explainable decisions.

B.2.1 *Monitor Agent*'s Behavior

The role of the *MA* is to be a proxy between *AV*s and the user. It interacts with the user via dialogues to build a user profile that simulates the user preferences and objectives. *MA* could be formed in a group of agents for fault tolerance and backup reasons and to avoid having a bottleneck in the model. Additionally, members inside this group may execute different explanations behaviors and interact/cooperate to provide the explanation to the human. The most important point is to have one interface with the human user to avoid overwhelming her/him with many interfaces. We can look at the monitor agent as the personal assistant of the human that could be embedded in his/her smartphone for example. Therefore, and even with a group of*MA*s, the interface with the human is preferably unified through one agent as a representative of the group.

MA gathers the statistics of decisions and their explanations from *AV*s. It aggregates these explanations in several abstraction levels. Following a similar approach of [127], the *MA* builds a multilevel explanation tree. The leaves of this tree correspond to particular agent's actions explanations. The root corresponds to the global abstract explanation for the final recommendation, and intermediate levels correspond to explanations for the evolution of evaluation metrics. At the end of the simulation scenario, it ranks the different solution methods based on their matching to the user profile providing a summary explanation for the ranking decision. The user could ask for a detailed explanation –to handle this, the *MA* defines a new granularity for selecting the right level of explanation that is communicated to the user. While the user

is asking for more details, *MA* proceeds from the root to leaves gradually, providing at every step the corresponding level of explanations. It stops when the user stops asking or reaching the leaves representing the atomic details that can not be expanded. The next section discusses how an *MA* computes its recommendation.

B.2.2 Computing the recommendations

The objective of *MA* is to assign values to its decision variable by the end of the scenario execution. *MA* has three sets of variables: profiles, recommendation and explanation variables. The recommendation variables are the ranking values for the different candidate methods. The explanation variables aggregate individual *AV*s' explanations and *MA*'s reasoning on the evolution of the evaluation metrics during the execution. The profile variables define a model based on the available features of allocation methods that match the user-defined features profile. If we manage to get such a model, then making recommendations for a user is relatively easy. We need to look at the user profile and compute its similarity to the different candidate methods. The candidates are then ranked based on their similarity value.

The user profile u in the set of user profiles U is represented by a vector of n features $u = [u_1, ..., u_n]$ defines the user's preferred values for the different evaluation metrics. Given that the system implements k solution methods that are potential candidates, this set of candidate methods is represented by $M = \{m_1, ..., m_k\}$; $m \in M$ is the feature vector of the candidate $m = [f_1^m, ..., f_n^m]$ we define a distance $dist$ function to calculate the n-dimensional euclidean distance between feature vectors:

$$dist : U \times M \rightarrow \mathbb{R}^+$$

$$dist(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

A perfect-match method m' to user profile u if exists, will have $dist(u, m') = 0$ otherwise the following similarity function will be used to rank the candidate methods:

$$sim : U \times M \rightarrow [0, 1]$$

In its simplest form, sim function is the inverse of $dist$.

$$sim(x, y) = \frac{1}{dist(x, y)}$$

so that the highest recommended method m' to user u is the one with higher value of $sim(u, m')$.

B.2.3 Creating and communicating the explanations

As seen before, we need explanations that are scalable for multiple levels, we can distinguish two types of actions to be explained: the *AV*'s individual decisions and the aggregated decision by *MA*.

The classical approach in XAI is the straightforward design of interpretable models on the original data to reveal the logic behind actions proposed by the system. State-of-the-art interpretable models, including decision trees, rules, and linear models; are considered to be understandable and readable by humans [112]. This applies to the individual decisions of *AV*s in our model, every *AV* is an autonomous agent having predefined interpretable behavior, and can justify his decisions with their technical and social reasons (based on its believes of itself and the context).

Another XAI approach is the post hoc interpretability, given the decisions made by the system, the problem consists of reconstructing an explanation to make the system intelligible

without exposing or modifying the underlying model internally. The generation of explanations is an epistemic and computational action, carried out on-demand according to the current state of a model, and meta-knowledge on the functionalities of the system. It is intended to produce a trustworthy model based on features or exemplars. This applies to the aggregation of decisions made by *MA*, to the statistics-based matching, and to the recommendations.

An explanation can indeed be in any type of interaction. The advantage of human-like interaction is that it provides to the user higher levels of satisfaction, trust, confidence, and willingness to use autonomous systems. For this reason, many techniques have been developed to generate natural language (NL) descriptions of agent behavior and detected outliers or anomalies. This entails answering questions such as, why did an agent choose a particular action? Or what training data were most responsible for that choice? The internal state and action representations of the system are translated into NL by techniques such as recurrent neural networks [51], rationale generation [52], adding explanations to a supervised training set such that a model learns to output a prediction as well as an explanation [32].

Appendix C

Synthèse des parties clés de la thèse

Parmi les questions importantes dans la gestion des systèmes de transport à la demande (ODT) nous retrouvons les problèmes d'allocation de demandes de courses aux véhicules qui soient réalisables et efficaces. Les taxis autonomes sont une catégorie de véhicule sans conducteur qui en plus de leur autonomie de déplacement dans le trafic ont l'autonomie du choix des clients à servir. Cela signifie qu'ils sont responsables de leur choix d'affectation aux demandes (en prenant des décisions décentralisées) ou d'exécuter les plannings qui sont décidés de manière centralisée par un répartiteur. En pratique, la faisabilité et l'efficacité du choix de centraliser/décentraliser la solution dépendent de la complexité du problème, de ses contraintes (e.g. la topologie du réseau et la structure de la demande) et de la dynamique de l'environnement (e.g. le trafic routier, les aléas de conduite).

Dans cette thèse:

1. Nous proposons AV-OLRA, un modèle générique pour le problème d'allocation des ressources en ligne avec des véhicules autonomes. Ce modèle est générique car il est indépendant des solutions mises en oeuvre. Ainsi nous modélisons les données du problème (composantes, contraintes) et les indicateurs permettant d'évaluer les différentes stratégies d'allocation ;

2. Nous proposons un modèle multi-agents générique support de solutions au problème des système ODT, où les véhicules autonomes (agents) communiquent avec leurs voisins via une communication pair-à-pair au sein d'ensembles connectés ; Ce modèle est générique car il supporte la réalisation des solutions en limitant les nouveaux développements aux particularités de la méthode d'allocation.

3. Nous classons les différentes méthodes d'allocation en fonction du comportement de coordination des agents;

4. Nous présentons un nouvel algorithme heuristique décentralisé d'insertion (ORNInA) par lequel les véhicules coordonnent leurs décisions via un système d'enchères. Le mécanisme de décision de cette approche est 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.

5. Nous évaluons et comparons expérimentalement différentes méthodes de résolution (optimisation linéaire en nombres entiers, auto-évaluation, enchères et optimisation sous contraintes distribuée).

Ce chapitre résume les plus importants éléments de cette thèse. Il est structuré comme suit. La section C.1 résume la partie I (état de l'art) en présentant quelques travaux relatifs à l'allocation de ressources multi-agents avec un focus sur les travaux concernant les systèmes ODT. Sur la base de cette analyse de l'état de l'art, nous exposons le problème AV-OLRA en section C.2, et un modèle multi-agents générique pour le résoudre dans la section C.3. La section C.4 examine plus en détail les différents mécanismes de coordination investigués dans cette étude, y compris ORNInA auquel nous consacrons la section C.5. Ces mécanismes de coordination sont ensuite évalués expérimentalement dans la section C.6.

C.1 Allocation de ressources multi-agents et système ODT

Ces dernières années, le nombre d'articles consacrés à l'application des technologies fondées sur les agents dans le domaine des transports a considérablement augmenté attestant les bénéfices de cette approche pour ce domaine d'application. [129] propose une synthèse des publications concernant des simulations et modèles pour systèmes ODT. Dans ce domaine, mieux répondre à la demande est considéré comme un défi et doit prendre en compte les personnes avec leur comportement et leur interaction avec un environnement de transport complexe. La vision MARA (*Multi-Agent Resource Allocation*) est pertinente pour la résolution des problèmes ODT et a été mise en oeuvre selon différentes approches. Ainsi, la centralisation du processus d'allocation avec un répartiteur automatique est encore assez courante dans les approches multi-agents [50, 102, 143]. D'autre part et pour réaliser une planification en temps réel des services ODT, plusieurs modèles décentralisés ont été proposés [62, 84]. Un modèle théorique de système de transport est développé dans [89] pour étudier le comportement de coopération des véhicules, avec une perspective globale ; la stratégie permettant d'obtenir la meilleure efficacité est de partager les informations entre véhicules coopérants dans un réseau de transport flexible (i.e les horaires et itinéraires sont variables). Dans le cas contraire (absence de communication entre les agents), [155] a proposé d'utiliser la programmation génétique pour développer des systèmes multi-agents décentralisés qui résolvent les problèmes dynamiques des systèmes ODT. Les auteurs ont conclu que la planification à long terme n'est pas bénéfique dans de tels contextes en raison de la très forte dynamique ; ainsi, les agents ne devraient examiner qu'une seule demande à l'avance.

L'un des principaux défis que pose l'utilisation des approches MARA et plus généralement multi-agents, pour résoudre les problèmes liés à l'ODT est le goulot d'étranglement en matière de communication. Une solution est une organisation spatiale des agents, e.g. une zone de planification limitée par agent et pas de communication [81]. Un second défi concerne la dynamique du système. Les solutions aux problèmes d'allocation de ressources pour un système ODT dans des environnements dynamiques doivent remettre en question les plannings des véhicules en temps réel. Cette remise en cause rend dans la pratique l'objectif de proposer une solution optimale inaccessible. Cependant, la conception d'approches itératives pour obtenir des solutions réalisables en temps raisonnable est une alternative appropriée pour aborder l'aspect dynamique ; cela nécessite de considérer le besoin de communication (pour mettre à jour les informations) et donc de fournir des schémas de communication et de coordination solides et efficaces.

De même, un cadre de modélisation générique, c'est-à-dire indépendant de la solution/stratégie de résolution, du problème d'allocation de ressources localisées en ligne (OLRA), et un système multi-agents pour résoudre le problème de la gestion du stationnement urbain ont été proposés dans [167]. La solution repose sur une communauté de conducteurs qui partagent leurs connaissances locales sur la disponibilité des places de stationnement. Notre travail s'appuie sur ce dernier modèle pour proposer un modèle spécifique au problème de l'allocation des ressources à la demande dans les flottes de véhicules autonomes.

C.2 Modèle de problème AV-OLRA

Dans ce travail, nous définissons le problème AV-OLRA comme une spécialisation du modèle OLRA pour l'allocation de ressources en ligne avec des véhicules autonomes, et une extension pour la prise en compte de la communication et la modélisation de contraintes temporelles supplémentaires.

Nous formulons donc le problème AV-OLRA comme suit :

$$\text{AV-OLRA} := \big(\mathcal{R}, \mathcal{V}, \mathcal{G}, \mathcal{T}\big) \tag{C.1}$$
$$\mathcal{R} = \{r_i | i \in \mathbb{N}\} \tag{C.2}$$
$$\mathcal{V} = \{v_i | i \in \mathbb{N}\} \tag{C.3}$$
$$\mathcal{G} = \big(\mathcal{N}, \mathcal{E}, \omega\big) \tag{C.4}$$
$$\mathcal{T} := \{t_0, t_1, \ldots, t_{end}\} \tag{C.5}$$

où $\mathcal{R}$ définit un ensemble dynamique de demande des passagers (les ressources dans le modèle OLRA) qui sont disponibles sur une fenêtre temporelle spécifique au moment de l'exécution ; l'ensemble des véhicules $\mathcal{V}$ (ensemble des consommateurs $\mathcal{C}$ dans le modèle OLRA) représente une flotte de m véhicules autonomes qui sont mobiles et ne peuvent communiquer qu'à une portée limitée ; $\mathcal{G}$ est un graphe dirigé, avec $\mathcal{N}$ l'ensemble des nœuds (carrefours), et $\mathcal{E}$ l'ensemble des arcs (routes), $e_{ij} \in \mathcal{E}$ est l'arc entre les nœuds i et j, ω est une fonction d'évaluation qui associe à chaque arc $e \in \mathcal{E}$ une valeur ω_e sur la base d'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. Enfin $\mathcal{T}$, est l'horizon temporel du problème.

Definition 5 *Un véhicule autonome $v \in \mathcal{V}$ est caractérisé par sa capacité (nombre maximum de passagers) c, son coût de trajet par distance parcourue cpd et une portée de communication limitée rng*

$$v := \big(c, cpd, rng\big)$$

ainsi qu'un ensemble de propriétés dépendantes du temps qui sont

- $\texttt{loc} : \mathcal{V} \times \mathcal{T} \to \mathcal{N} \cup \mathcal{E}$ *sa localisation actuelle,*
- $\texttt{dest} : \mathcal{V} \times \mathcal{T} \to \mathcal{N}$ *sa destination actuelle,*
- $\texttt{seats} : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+$ *le nombre de sièges actuellement disponibles,*
- $\texttt{dist} : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+$*, la distance parcourue depuis* t_0
- $\texttt{distR} : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+$*, la distance parcourue avec un client depuis* t_0
- $r_k : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+$*, le nombre de demandes connues depuis* t_0
- $r_s : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+$ *le nombre de demandes satisfaites depuis* t_0
- $\texttt{messageC} : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+ \times \mathbb{N}^+$ *le nombre de messages envoyés depuis* t_0
- $\texttt{messageS} : \mathcal{V} \times \mathcal{T} \to \mathbb{N}^+ \times \mathbb{N}^+$ *le volume de messages envoyés depuis* t_0 *.*

La communication entre deux composants du système est réalisée si la distance qui les sépare est inférieure ou égale à leur portée de communication. Cependant, comme la portée de communication des véhicules est limitée, et pour maximiser leur connectivité, deux véhicules peuvent être connectés par transitivité. Cela conduit à la définition suivante d'un ensemble connecté :

Definition 6 *Un ensemble connecté (noté CS pour connected set) est un ensemble d'entités connectées directement ou par transitivité.*

Les CSs sont des entités dynamiques ; ils sont créés, divisés, fusionnés en cours d'exécution en fonction du mouvement des véhicules. Ainsi, selon la définition précédente, un véhicule ne peut communiquer au temps t qu'avec les membres de son CS par des messages directs ou diffusés. De même, il ne peut recevoir que les demandes accessibles au sein de l'ensemble connecté selon les mêmes règles de communication. La portée de communication limitée divise implicitement la flotte en plusieurs ensembles connectés.

Definition 7 *Une solution à un problème AV-OLRA est définie pour chaque ensemble connecté comme une agrégation des* allocations *de tous les* consommateurs *de cet ensemble qui évite tous les conflits.*

Cette définition implique qu'une solution à un problème AV-OLRA défini pour des véhicules et des demandes peut être sous-optimale parce que la solution optimale n'est pas l'union des sous-solutions optimales de chaque CS. En outre, toute solution dépend du temps à cause de l'aspect *en ligne* du problème.

La portée de communication dépend de la technologie de communication utilisée (formalisée par la propriété *rng* de vehicule) et de la densité du réseau (propriété de transitivité de formation des ensembles connectés). Considérer cette dimension dans notre modèlisation du problème contribue à sa généricité en permettant de considérer son impact sur différents types de solution. Ainsi, plus la portée de communication est faible, plus il existe d'ensembles connectés ; cela signifie que pour une communication à courte portée, même avec des approches centralisées, le calcul de la solution est décentralisé vers plusieurs répartiteurs. Une flotte ayant une portée de communication suffisamment longue pourrait revenir à un seul ensemble connecté à l'échelle d'une ville avec un partage global des connaissances. Une approche centralisée conduit alors à un répartiteur central pour déterminer la solution globale.

La qualité d'une allocation est caractérisée par des indicateurs fonctionnels et techniques dont le calcul est indépendant des approches de résolution mais qui permet de comparer leur faisabilité et leur qualité. Les indicateurs fonctionnels sont des mesures de l'optimalité du processus d'allocation défini par sa fonction objectif, tandis que les indicateurs techniques sont utilisés pour évaluer la faisabilité et l'applicabilité du processus d'allocation et pour prévoir ses coûts opérationnels dans différents contextes.

Dans ce travail, nous caractérisons la qualité d'une solution AV-OLRA dans les scénarios ODT par les indicateurs suivants calculés à partir de la définition des véhicules autonomes :

Quality est le pourcentage de demandes satisfaites (consommées) sur toutes les demandes annoncées. Par conséquent, cet indicateur indique le niveau de qualité de service (QoS). Son calcul repose sur l'utilisation des propriétés k et r

Utility est l'*utilité totale* des plannings des véhicules, dérivée des distances des voyages réalisés (effectués avec un passager à bord, de la source à la destination $distR$), qui définit le **gain** pour l'entreprise.

Cost est le *coût opérationnel*, dérivé des distances totales parcourues par les véhicules $dist$. La relation entre les indicateurs **Utility** et **Cost** définit la qualité des affaires (QoB).

MsgCount est le *nombre total de messages* échangés au cours du processus d'allocation dérivé de $messageC$.

MsgSize est la *taille moyenne des messages* échangés pendant l'allocation dérivé de $messageS$.

Ces deux derniers indicateurs de communication estiment le coût technique de la solution et permettent de prédire si elle est applicable en termes de charge de communication, c'est-à-dire si elle pourrait provoquer des goulets d'étranglement critiques. Ces indicateurs sont génériques car fondée sur l'utilisation de données issues de des activités élémentaires des véhicules.

C.3 Approche multi-agents pour AV-OLRA

Dans cette section, nous décrivons notre modèle multi-agents support aux solutions au problème AV-OLRA. L'environnement du problème AV-OLRA représente la topologie de l'infrastructure urbaine $\mathcal{G}$ et le modèle de communication des agents tel que décrit par la définition 6.

Il n'y a qu'un seul type d'agents dans notre modèle. Un agent véhicule autonome (AV) est associé à chaque véhicule du système. Nous pouvons distinguer trois sous-comportements différents (*acting*, *communicating* et *planning*). Comme nous modélisons AV-OLRA dans un espace de temps discret, l'horizon temporel est défini comme un ensemble de pas d'exécution. À chaque pas, chaque agent effectue les actions suivantes, comme le montre la figure 5.3 à la page 48 :

1. lire les messages reçus et mettre à jour le contexte (sous-comportement de communication, *communicating*) ;
2. choisir les lieux à visiter (sous-comportement de planification, *planning*) ;
3. agir en effectuant une action de conduite (sous-comportement d'action, *acting*) ;
4. diffuser ses informations contextuelles (sous-comportement de communication, *communicating*).

Le sous-comportement de planification, *planning* et ses relations aux autres sont à spécialiser pour chaque solution (représentation en pointillé figure 5.3) alors que les autres seront identiques pour chaque solution.

C.3.1 Sous-comportement d'action

En fonction de la présence de passagers à bord, de la localisation du véhicule et de sa connaissance des demandes à venir, une agent AV peut se trouver dans l'un des états suivants (figure 5.4 à la page 49) :

Marauding : le véhicule n'a pas de passager à bord et cherche sa prochaine destination ;

Moving : le véhicule a une destination et sy rend selon la topologie urbaine ;

Picking up : le véhicule est à l'emplacement d'origine de la demande du passager p afin d'effectuer l'action `pick_up`(p) avant de reprendre son déplacement ;

Dropping off : le véhicule est à l'emplacement de destination de la demande du passager p afin d'effectuer l'action `drop_off`(p) avant de rechercher une nouvelle destination.

Les transitions entre ces états sont illustrées dans la figure 5.4.

C.3.2 Sous-comportement de communication

En tant qu'agents communicants, les AV ont un comportement de communication avec les autres entités environnantes ; ils peuvent rejoindre/quitter des ensembles connectés (CS), diffuser, envoyer et recevoir des messages.

- `join`(cs) : l'agent rejoint le CS cs du fait qu'il se trouve dans le rayon de communication d'au moins un de ses membres ;
- `leave`(cs) : l'agent quitte le CS cs car il n'est plus dans le rayon de communication d'au moins un membre ;
- `send`(m, a) : l'agent envoie un message m à un autre agent a à condition qu'ils soient dans le même CS ;
- `receive`(m) : l'agent reçoit un message m d'un autre agent de son CS (une fois reçu et lu, le message est stocké dans la base de croyances de l'agent) ;
- `broadcast`(m) : similaire à `send`(m, a) mais ici l'agent ne spécifie pas l'agent récepteur, il diffuse plutôt le message à l'ensemble des membres de son CS.

C.3.3 Sous-comportement de planification

Le comportement des AVs en matière de planification dépend du mécanisme d'allocation choisie (centralisée/décentralisée, coopérative/compétitive, avec/sans modèle de coordination). La figure 5.5 à la page 51 illustre le comportement de planification abstrait et générique des AVs. Pour mettre à jour son planning, un AV recherche en permanence des options de planification. Si une option est trouvée, l'AV en sélectionne une et communique (ou pas selon le modèle de coordination) sa décision à ses voisins (les autres agents de son CS). Le voisinage parvient à un accord ou à un désaccord, selon le mécanisme de coordination et l'option choisie. En cas d'accord, l'AV met à jour son planning et recherche l'option suivante et ce, jusqu'à ce qu'aucune option ne soit disponible. La nature des options de planification dépend également du mécanisme de coordination.

Bien que nous soutenions plusieurs modèles de coordination, nous souhaitons étudier chacun individuellement et non les conséquences de leurs interactions. Par conséquent, nous considérons que l'ensemble des agents de la flotte est homogène. Dans la section C.4, nous présentons en détail différents mécanismes de coordination que nous utilisons pour valider notre modèle, y compris certaines approches coopératives avancées comme l'usage d'algorithmes DCOP et un mécanisme d'enchères pour le comportement de coordination des agents. Ce dernier répond aux exigences du problème AV-OLRA en fournissant une solution utilisant des calculs légers, dynamiques et continuellement sujets à amélioration.

C.3.4 Utilité, contraintes et objectif

Dans tout problème MARA, la fonction d'utilité représente le degré de satisfaction d'un agent pour une allocation donnée [28]. Chaque agent a une valeur d'utilité exprimée sous la forme d'une valeur explicite ou d'une relation qui révèle la solution la plus satisfaisante (optimale). Une procédure d'allocation tente de fournir aux agents des ressources qui correspondent autant que possible à leur exigence. Dans notre modèle, nous définissons la fonction d'utilité des agents AV sur la base des indicateurs de la qualité de solution décrits dans la section C.2. Nous considérons que plus un agent satisfait de demandes, plus il doit gagner en valeur d'utilité. Ainsi, l'indicateur **Quality**, s'il est considéré individuellement pour chaque agent $a \in \mathcal{V}$, définit

son utilité :

$$u_a = \frac{r_s(a, t_{end})}{r_k(a, t_{end})} \tag{C.6}$$

Bien entendu, l'attribution de la demande r à l'agent a est contrainte par la disponibilité spatiale et temporelle de a et de r. Nous considérons que l'origine et la destination de chaque demande sont des constantes, et qu'une demande n'est disponible pour être prise en charge qu'à son origine pendant sa fenêtre temporelle définie $w_r[l_r, u_r]$. Ainsi, l'affectation de r à a exige que a puisse arriver au point d'origine de r à un moment t situé entre la limite inférieure l_r et la limite supérieure u_r de la validité temporelle de r.

Dans ce document, nous n'envisageons pas de scénarios de partage de véhicules. Nous supposons qu'un trajet en véhicule est consacré à une seule demande, mais nous devons tout de même tenir compte de la taille de la demande (nombre de sièges requis) et de la capacité du véhicule. Cela implique que la définition de la disponibilité doit également inclure la contrainte de capacité :

$$\exists t \in w_r[l_r, u_r] : \texttt{seats}(a, t) \geq s_r \ \ \& \ \ \texttt{loc}(a, t) = o_r \tag{C.7}$$

Le fait d'être membre d'une flotte impose aux AV d'être coopératifs et de suivre le mécanisme de coordination prédéfini pour atteindre leur objectif global. Dans une perspective globale d'ODT en tant que modèle commercial, l'objectif principal des prestataires de services ODT est de gagner la satisfaction des utilisateurs. Cela signifie que leur objectif est de réduire les coûts et d'augmenter les gains. De ce point de vue, nous pouvons définir la fonction objectif $\mathcal{F}$ à maximiser par le processus d'allocation fondé sur la relation entre les indicateurs **Utility** et **Cost** :

$$\mathcal{F} = \sum_{v \in \mathcal{V}} (P + p * \texttt{dist}(v, t_{end})) - \sum_{v \in \mathcal{V}} cpd_v * \texttt{distR}(v, t_{end}) \tag{C.8}$$

P est un prix fixe (frais de service) par demande, p est un facteur de tarification par unité de distance parcourue.

C.4 Mécanismes de coordination

Cette section illustre certains comportements de coordination que les flottes de véhicules suivent habituellement pour atteindre un objectif global d'allocation. Pour chacun, nous présentons le modèle de coordination correspondant. Un mécanisme de coordination est défini par trois composantes $\langle DA, AC, AM \rangle$, où DA indique le niveau d'autonomie de décision qui est soit centralisé (C) soit décentralisé (D) ; AC indique s'il y a coopération des agents en notant S s'ils coopèrent et partagent des informations sur les plannings et N s'il n'y a pas de coopération. AM est le nom du processus d'allocation.

C.4.1 Comportement égoïste

Le mécanisme de coordination noté $\langle D, N, \text{Greedy} \rangle$ est fondé sur un processus d'allocation décentralisé avec des agents compétitifs et sans coordination explicite. Dans ce mécanisme, les agents ne s'appuient pas sur les décisions des autres et n'échangent jamais leurs plans. Dans les scénarios du monde réel, une stratégie de ce modèle est fondée sur l'avidité, dans laquelle le véhicule ne considère qu'une seule demande à l'avance et la meilleure pour lui (par exemple, la plus proche, afin de raccourcir la distance de conduite à vide) [155]. Lorsqu'un véhicule ne transporte pas déjà des clients, il doit décider quelle demande il traitera en premier, en fonction des informations dont il dispose sur les demandes disponibles. Une heuristique calcule une valeur de priorité pour chaque demande. Ensuite, l'agent traite en premier la demande ayant la valeur de priorité la plus élevée. Des conflits peuvent survenir, mais ils sont résolus simplement en

appliquant la politique du " premier arrivé, premier servi "., Alors, si l'agent n'a pas de passager à bord ses options sont ses demandes connues et réalisables, sinon aucune option n'est prise en compte. L'état *coordinating* est ignoré (comme s'il parvenait à un accord pour toute option choisie), de sorte que la qualité de la solution dépend de la stratégie de l'agent pour choisir la demande suivante.

C.4.2 Comportement avec répartiteur

Ici, le mécanisme de coordination est centralisé, le rôle de l'agent est de mettre à jour son planning en fonction de ce qu'il reçoit du répartiteur. $\langle C, S, \text{MILP}\rangle$ est un exemple de ce type de mécanisme avec la résolution MILP (Mixed-Integer Linear Problem) comme processus d'allocation. Dans notre modèle, nous avons besoin d'un répartiteur par ensemble connecté (CS). Ainsi, lors de la création (ou de la mise à jour), un membre d'un CS (par exemple, celui qui a l'indice le plus bas dans l'ensemble) devient le répartiteur qui sera responsable de collecter les informations des autres agents et sur demande. Il doit aussi faire le calcul d'allocation par lui-même, ou en appelant un service externe pour obtenir une allocation optimale (résolvant un MILP), puis envoyer à chaque autre véhicule son planning potentiel, comme dans [53, 91, 162]. Dans ce modèle centralisé, le rôle du comportement de planification des AVs est de demander à l'agent responsable (sur un portail ou un véhicule) de mettre à jour leur planning en permanence. Dans ce cas, la seule option disponible est de requêter le portail, et l'état *coordinating* consiste en un protocole de demande/réponse qui enverra le nouveau planning sous forme d'accord.

C.4.3 Comportements réellement coordonnés

Dans ce cas, de type $\langle D, S, PC\rangle$, le mécanisme de décision est décentralisé, les agents sont coopératifs et un protocole de coordination (PC) est appliqué pour l'allocation. Dans cette catégorie de mécanismes de coordination, les agents échangent des informations et coopèrent pour atteindre un objectif commun, en évitant les conflits et en optimisant la qualité de la solution. Il existe plusieurs approches pour atteindre ce comportement, comme l'optimisation sous contraintes distribuées (DCOP) [56], les protocoles de négociation [84, 50] et les enchères [120]. Nous instancions ici des solutions par enchères et par DCOP.

Coordination par enchères. Les enchères sont très courantes dans les situations quotidiennes et fournissent une base conceptuelle générale pour comprendre les problèmes d'allocation des ressources au sein d'ensembles d'agents [144]. Nous présentons ici un exemple de mécanisme collaboratif de construction de plannings de véhicules, dont notre proposition ORNInA (voir [41]), noté $\langle D, S, \text{Auction}\rangle$, fondé sur des enchères pour coordonner de manière pair-à-pair les décisions de planification de flottes de véhicules autonomes. Ce mécanisme est proposé pour fonctionner dans un cadre dynamique, entre des agents véhicules qui appartiennent à un ensemble connecté dans lequel ils peuvent recevoir et envoyer des messages directs ou diffusés. Les agents intéressés par une demande donnée lancent des enchères au premier prix pour cette demande, et le gagnant l'ajoute à son planning. La détermination du gagnant est un processus complètement décentralisé. Afin d'améliorer l'efficacité de la planification dans des contextes dynamiques, les agents sont autorisés à échanger leurs demandes planifiées au moment de l'exécution, avec des tours d'enchères supplémentaires pour décider si cet échange augmente la valeur de la fonction objectif au sein du CS. Les agents communiquent entre eux par des messages directs ou indirects pour partager des informations ou coordonner leurs décisions.

Coordination par DCOP. Dans $\langle D, S, \text{DCOP}\rangle$, les agents décident seuls mais se coordonnent avec les agents du même ensemble connecté en utilisant un algorithme d'optimisation sous contraintes distribuée afin d'éviter les conflits au sein du CS. À chaque fois qu'un ensemble connecté change, un DCOP $P = \langle A, X, D, C\rangle$ est généré à partir de l'instance AV-OLRA pour maximiser la fonction objectif dans l'équation C.8, comme suit. A définit l'ensemble des

agents dans l'ensemble connecté. X définit l'ensemble des variables de décision dans trois sous-ensembles (x_{ij}'s, y_{ij}'s et z_{ij}'s) : $x_{ij} \in X$ est une variable binaire égale à 1 si le véhicule v_i sert la requête r_j ; y_{ij} est une variable binaire égale à 1 seulement si la requête r_j est la première requête à être servie par v_i. Enfin, z_{ij} est une variable entière qui définit à quel moment une requête r_j est visitée par v_i. D définit les domaines des variables : $\{0, 1\}$ pour les x_{ij} et les y_{ij}, et un ensemble de domaines de plages de temps définissant la fenêtre $[l_j, u_j]$ pour chaque z_{ij}. C définit l'ensemble des contraintes, qui se compose de contraintes dures (capacité, disponibilité spatio-temporelle et fenêtres temporelles) et de contraintes souples définissant le coût et l'utilité de la décision d'allocation (utilisées pour calculer la valeur de la fonction objectif).

Les algorithmes DCOP sont variés, et le choix dépend de l'objectif de la solution et du contexte du problème. Les caractéristiques d'exécution de l'algorithme (temps d'exécution, nombre/taille des messages et besoin en mémoire par agent) sont des facteurs essentiels pour traiter les problèmes dynamiques en ligne.

C.5 ORNInA: L'heuristique d'insertion et amélioration continue par enchères

Dans cette approach, nous utilisons une heuristique d'insertion comme celle décrite par Solomon [147] 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 [39].

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.

En considérant le coût opérationnel d'un voyage comme la longueur totale de son trajet. 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.

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. Dans ce qui suit, nous proposons un protocole d'optimisation locale pour améliorer la qualité de la solution. Ce protocole est déclenché à chaque cycle de simulation au cours duquel les véhicules d'un ensemble connecté peuvent échanger des parties de leur planification. Les étapes d'exécution de ce protocole sont :

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

Étape 2 Chaque demande 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.

À l'instar de la stratégie *Rolling Horizon* de Agatz et al. [2], 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. [2], 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.

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 $\mathcal{O}(n^3)$[22]. 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.

C.6 Évaluation expérimentale

Dans cette section, nous présentons les résultats expérimentaux de l'instanciation du modèle AV-OLRA avec le modèle multi-agents décrit dans la section C.3, en prenant en charge les différents types de mécanismes de coordination de la section C.4. Le modèle est mis en œuvre en tant que système multi-agents avec un simulateur de transport en temps discret.

C.6.1 Cadre expérimental

Nous explorons deux types de scénarios pour nos expérimentations. Le premier, sur données synthétiques, est fondé sur le réseau routier de la ville de Saint-Étienne. Plus de 1400 arcs ont été extraits de Open Street Map (OSM)[1] et post-traités pour un quartier situé entre les coordonnées GPS (45.4325,4.3782) et (45.437800,4.387877) pour produire un graphe formé de 71 arcs. Les demandes des passagers sont générées aléatoirement avec des lieux de ramassage et de dépôt appartenant à un ensemble spécifique de lieux appelé *sources*. 40 lieux répartis uniformément sur la carte ont été sélectionnés comme sources pour l'émission de la demande.

Le second scénario est fondé sur le réseau urbain en forme de grille de l'arrondissement de Manhattan dans la ville de New-York. Pour ce scénario, les demandes des passagers ont été extraites de l'ensemble de données de NYC-TLC[2].

L'idée principale qui motive l'utilisation de deux types de scénarios est d'expérimenter notre modèle SMA et la performance des approches de solution dans une variété de situations. Les scénarios de données synthétiques permettent de contrôler la taille du problème et la distribution de la demande lors de la création des instances du problème. Ils sont donc idéaux pour diagnostiquer les paramètres symptomatiques et pour régler les paramètres des algorithmes. D'autre part, l'extraction d'instances à partir de jeux de données tels que NYC-TLC permet d'expérimenter la performance des solutions sur des instances plus grandes (avec des tailles de flotte plus importantes et une plus grande densité de demandes) sur la base d'une distribution similaire au déploiement dans le monde réel.

Lorsque les véhicules doivent échanger directement des messages, nous considérons qu'ils communiquent via DSRC[3] avec une portée de communication réaliste de 250 mètres. Le nombre de demandes générées et le nombre de véhicules sont des paramètres de la simulation. Tous les scénarios ont une durée de 1000 cycles et, à chaque cycle, 0 ou 1 demande est générée. Nous évaluons la performance des 5 mécanismes de coordination vus en section C.4 dont celui fondé sur un DCOP avec l'algorithme DSA, variante A, $p = 0.5$ [169] $\langle D, S, \text{DCOP}(\text{dsa})\rangle$, et celui fondé sur un DCOP avec l'algorithme MGM-2 [121] $\langle D, S, \text{DCOP}(\text{mgm-2})\rangle$.

Le système multi-agents et le simulateur hébergé par la plateforme Territoire[4], implémentés en Java, ont été exécutés sur un processeur Intel® Xeon® E-2146G CPU @ 3.50GHz, avec 32 Go de RAM DDR4. Les algorithmes DCOPs ont été mis en œuvre en utilisant la bibliothèque FRODO [90].

C.6.2 Le choix et le paramétrage des algorithmes de DCOP

Compte tenu des caractéristiques de notre problème, une comparaison (sur données synthétiques) des algorithmes DCOP en termes de charge de communication et de temps d'exécution en fonction de la taille du problème est présentée dans la Figure 10.1 à la page 103. Cette comparaison a permis d'éliminer ceux qui nécessitent un temps d'exécution trop important et/ou une charge de communication trop élevée.

ADOPT, DPOP, ASO-DPOP, SynchBB et AFB sont connus pour être des algorithmes complets qui trouvent des solutions optimales mais ont une complexité algorithmique exponentielle. D'autre part, Max-Sum est un algorithme incomplet avec un temps d'exécution raisonnable,

[1] urlhttps://www.openstreetmap.org

[2] The New York City Taxi and Limousine Commission (NYC-TLC) collecte des données, telles que les enregistrements des trajets, le nombre de véhicules et les tarifs dans la ville de New York sous forme de fichiers CSV par mois et par type de véhicule (taxis jaunes, taxis verts, véhicules de location et véhicules de location à gros volume)

[3] La communication de véhicule à véhicule via la communication dédiée à courte portée (DSRC) offre une connectivité réseau rapide et à faible latence dans un rayon de communication allant jusqu'à 300 mètres.

[4] territoire.emse.fr

mais qui nécessite une taille de mémoire exponentielle par rapport au nombre moyen de voisins de l'agent dans le graphe de contraintes, c'est-à-dire une mémoire exponentielle par rapport à l'arité des contraintes.

Max-Sum opère sur un graphe de facteurs qui est un graphe biparti dans lequel les variables et les contraintes sont représentées par des nœuds. Chaque nœud représentant une variable dans le DCOP est connecté à tous les nœuds de fonction qui représentent les contraintes (ou facteur) dans lesquelles elle est impliquée. De même, un nœud de facteur est connecté à tous les nœuds de variable qui représentent les variables dans le DCOP original qui sont incluses dans la contrainte qu'il représente. Chaque agent adopte le rôle du nœud représentant sa propre variable et le rôle d'un des nœuds de fonction représentant une contrainte dans laquelle il est impliqué. Ainsi, dans notre cas avec les instances fortement contraintes, Max-Sum nécessite une grande quantité de mémoire pour la charge de communication et peut ne pas converger car le graphe factoriel comprend des cycles de tailles différentes.

Les autres sont les algorithmes DCOP de recherche locale dont la structure générale est synchrone. À chaque étape de l'algorithme, un agent envoie son affectation à tous ses voisins dans le réseau de contraintes et reçoit l'affectation de tous ses voisins. Ils diffèrent par la méthode que les agents utilisent pour décider s'ils doivent changer leurs affectations de valeurs actuelles à leurs variables. Par exemple, dans l'algorithme MGM (*Maximum Gain Message*), l'agent qui peut améliorer le plus son état dans son voisinage remplace son affectation. Dans MGM-2, la première étape consiste à décider quel sous-ensemble d'agents est autorisé à faire des offres. Chaque agent génère un nombre aléatoire de manière uniforme parmi $[0, 1]$ et se considère comme un offrant si le nombre aléatoire est inférieur à un seuil q. Si un agent fait une offre, il ne peut pas accepter les offres des autres agents. Tous les agents qui ne font pas d'offre sont considérés comme des récepteurs. Chaque récepteur choisit aléatoirement (uniformément) un voisin et lui envoie un message d'offre, qui consiste en tous les mouvements coordonnés entre l'offreur et le récepteur qui apporteront un gain d'utilité locale au récepteur dans le contexte actuel. La décision stochastique de remplacer une affectation est inspirée de l'algorithme stochastique distribué DSA. Dans DSA un agent génère un nombre aléatoire à partir d'une distribution uniforme sur $[0, 1]$ et agit si ce nombre est inférieur à un certain seuil p. Une valeur seuil plus faible réduit le nombre d'agents qui peuvent agir à chaque cycle, ce qui signifie une charge de messages plus faible et une qualité de solution légèrement inférieure, tandis qu'un seuil plus élevé signifie que chaque agent a plus de chances d'agir à chaque itération, ce qui signifie plus de cycles d'amélioration de la solution avec leurs dépenses en termes de communication.

Une comparaison (sur données synthétiques) de la qualité des solutions des algorithmes de recherche locale DCOP est présentée dans la Figure 10.2 à la page 104. Cette comparaison montre que les différentes variantes de DSA ont des performances presque identiques sur nos instances lorsqu'elles ont la même valeur p. Pour les variantes DSA et MGM, la qualité des solutions augmente avec le temps d'exécution de chaque cycle jusqu'à atteindre un point de stabilisation après lequel l'amélioration devient très faible. En fixant $p = 0.5, q = 0.5$, les algorithmes DSA et MGM-2 atteignent ce point de stabilisation après environ les cycles 30 à 50 itérations. Plus précisément, avec la variante A de DSA, le point de stabilisation de 80% est atteint en 45 itérations lorsque $p = 0,5$, et MGM-2 atteint ses 82% en 40 itérations lorsque $q = 0.5$. Sur la base de ces résultats, nous limitons nos expérimentations aux seuls algorithmes DCOP MGM-2 ($q = 0,5$) et DSA (variante A, $p = 0,5$).

C.6.3 Extraction d'instances AV-OLRA à partir de jeux de données NYC-TLC

À New York, il y a plus de 13 000 taxis pour satisfaire l'énorme densité de demandes. Ainsi, si nous voulons obtenir une qualité de service élevée dans nos expériences avec n'importe quel mécanisme d'allocation, nous avons besoin d'une flotte de taille similaire. Cependant, faire

une simulation microscopique sur des flottes de cette taille demande beaucoup de ressources. Par conséquent, nous devons produire des instances de taille inférieure qui restent toutefois représentatives de ces données. Pour ce faire, nous avons étudié le modèle statistique des données et, dans ce qui suit, nous présentons quelques analyses de données sur un fichier CSV provenant de l'ensemble des données d'enregistrement des trajets de NYC-TLC pour le mois de janvier 2018.

Sur la base du modèle statistique de jeux de données NYC-TLC, nous avons produit des instances de taille inférieure d'enregistrements de trajets qui contiennent une distribution spatiale et temporelle similaire des demandes, mais avec une densité inférieure. Pour chaque instance, nous avons choisi un nombre limité de zones, bien distribuées géographiquement dans Manhattan, et dont la densité de demandes varie. Puis nous avons sélectionné dans les fichiers CSV originaux un sous-ensemble de demandes courtes entre ces zones ayant des heures de prise en charge et de dépose correspondant à la distribution temporelle des données originales. Les figures 10.10, 10.12 et 10.13 aux pages 111, 113, et 114 illustrent une comparaison de la distribution spatiale et temporelle des demandes dans l'une des instances réduites et les données originales. Si nous ne considérons que Manhattan, les exemples réduits sont bien représentatifs des données originales sur les trajets courts, malgré le fait que le nombre de zones de prise en charge et de dépôt soit limité. Nous avons dans ces exemples des zones qui varient (de la même manière que les données originales) de zones très encombrées comme Times Square à des zones très peu encombrées en périphérie. Nous avons également le même schéma d'heures de pointe.

C.6.4 Resultats

Les figures 10.14 et 10.15 à la page 115 illustrent les performances des cinq approches en termes d'indicateurs de qualité de service (QoS) et de qualité des affaires (QoB) avec une analyse comparatives. Chaque point de ce diagramme représente la valeur de l'indicateur agrégée sur 1000 cycles de simulation. Ces deux figures montrent comment la qualité des solutions évolue avec l'augmentation de la taille de la flotte. Nous pouvons remarquer l'augmentation de la QoS et de la QoB avec l'augmentation du nombre de véhicules dans la flotte jusqu'à atteindre un point de bascule, après lequel il n'est plus possible d'améliorer la QoB en ajoutant des véhicules supplémentaires et la QoS n'est amélioré qu'à la marge.

Les valeurs obtenues par le comportement *avec répartiteur* représentent une limite supérieure pour la QoB car le répartiteur central calcule pour chaque cas la solution localement optimale compte tenu du contexte de l'ensemble connecté. Les performances des quatre autres approches varient selon les indicateurs. Ainsi, si Enchères domine DCOP DSA-A (QoS et QoB), sa domination sur DCOP MGM dépend du nombre de véhicules selon les indicateurs.

Étant fondé sur un algorithme glouton, dont le temps d'exécution est linéaire par rapport à la taille du problème, l'approche *égoïste* est très efficace en termes de temps de prise de décision. La raison en est qu'elle ne nécessite pas beaucoup de calculs pour sélectionner la demande la plus proche. L'inconvénient est ici l'ignorance de ce qui précède les décisions prises et le fait que les conflits de décisions entre différents véhicules (par exemple, deux véhicules vont vouloir prendre le même passager) ne sont résolus que tardivement, ce qui réduit la QoS. Avec un faible nombre de véhicules, les ensembles connectés sont peu nombreux et, par conséquent, la quantité d'informations partagées est réduite.

Les performances des approches coopératives et de l'approche avec répartiteur dépendent fortement de la quantité d'informations. Ayant des quantités similaires d'informations partagées, la qualité des quatre approches ne semble pas être très différente.

Avec des flottes de taille plus importante, davantage d'informations sont partagées dans les ensembles connectés. En outre, les véhicules passent plus fréquemment d'un ensemble connecté à un autre. Les trois approches coopératives ont des performances presque similaires. Pour

atteindre les mêmes valeurs de QoS avec une approche égoïste, il faut davantage de véhicules dans la flotte.

Si l'on considère les algorithmes DCOP (DSA et MGM-2), au temps t, et en partant d'une solution réalisable s_{t-1} obtenue au temps $t-1$, chacune de ces approches tente d'obtenir une nouvelle solution s_t dans laquelle la valeur de la fonction objectif est améliorée. Les algorithmes DCOP se concentrent sur la maximisation de l'utilité individuelle des agents pour maximiser l'objectif global, ce qui signifie attribuer aux agents des demandes qui augmentent son gain. Plus les demandes de voyage (potentiellement réussies) assignées à un agent sont nombreuses, plus l'utilité individuelle est élevée. Bien sûr, cela augmentera la QoB, mais l'effet principal sera sur la valeur de la QoS. D'autre part, chaque agent dans ORNInA, à chaque pas de temps, essaie d'ajouter à son planning au maximum une demande qui peut améliorer son utilité et donc les valeurs de QoS et QoB, puis la phase d'amélioration du planning, définie par le protocole pull-demand, essaie de réallouer les demandes de manière à maximiser le gain global même si cette réallocation diminue le gain de l'agent qui abandonne la demande. Le protocole pull-demand n'affecte que la valeur de la QoB car le nombre de demandes programmées (à servir) ne change pas, elles sont seulement réaffectées à d'autres agents. Ainsi, alors que la coordination basée sur les enchères d'ORNInA est plus performante que les algorithmes de recherche DCOP en termes de QoB, elle est surpassée par les deux en termes de QoS.

Pour atteindre un certain niveau de QoS, il faut un nombre minimum de véhicules, ce nombre varie selon les différentes approches, par exemple servir 90% des demandes[5] (illustré par la ligne noire pointillée sur la figure 10.15) nécessite de manière optimale 240 véhicules avec l'approche *Répartiteur*, tandis que ce nombre augmente à plus de 350 pour l'approche *Egoïste*. A la page 116, les figures 10.18 comparent les cinq approches en termes de taille de flotte requise pour atteindre cette QoS de 90%. La figure 10.18b montre que le mécanisme *Egoïste* nécessite un nombre supplémentaire de véhicules qui est presque le double de DCOP et 3 fois de *ORNInA*. (concernant l'optimal pour le Dispatching).

Le tableau 10.1 à la page 108 présente des indicateurs liés à la communication obtenus en simulant un scénario sur 1000 cycles, avec 10 véhicules, pour les différents comportements étudiés. Ici, les deuxième et troisième colonnes indiquent la taille maximale et moyenne des messages échangés (en octets) représentant l'indicateur **MsgSize**. La quatrième colonne rapporte l'indicateur **MsgCount** en termes de nombre moyen de messages reçus par un agent par cycle de simulation.

Même sans coordination, les agents échangent des messages d'information sur les nouvelles demandes annoncées. Ce type de message dépend du mécanisme de coordination. De nouveaux types de messages sont utilisés dans le mécanisme avec répartiteur : les messages de requête et de réponse échangés entre les véhicules et le répartiteur central. Les messages de requêtage contiennent simplement le contexte global de l'ensemble des véhicules connectés qui demandent au répartiteur de construire leurs plannings. Les messages de réponse sont envoyés par le répartiteur aux véhicules de manière individuelle et contiennent le planning potentiel de chacun. Ces messages peuvent être volumineux, en fonction de la taille du sous-problème. Les messages d'offre et de réponse utilisés par le mécanisme de coordination basé sur les enchères sont légers, de sorte que les valeurs de l'indicateur **MsgSize** restent proches de l'approche égoïste, tandis que la valeur **MsgSize** devient polynomiale dans le nombre d'agents dans l'ensemble connecté et le nombre de leurs demandes connues. Dans les deux mécanismes de coordination basés sur un DCOP (DSA et MGM-2), les agents d'un ensemble connecté instancient un DCOP entre eux chaque fois qu'ils doivent décider d'une mise à jour du planning. L'obtention d'une solution par l'un de ces algorithmes nécessite l'échange d'un grand nombre de messages. Ces deux algorithmes ne sont pas complets, ce qui signifie qu'ils poursuivent leurs essais pour améliorer la solution jusqu'à atteindre le *timeout* ou un optimum (local). Ceci accroît le nombre de messages

[5]En pratique, servir 90% des demandes représente une faible QoS. Cependant, ce seuil est choisi à titre d'exemple à des fins de comparaison, la même comparaison peut être effectuée avec n'importe quel seuil supérieur ou inférieur

échangés. D'autre part, la taille des messages échangés par ces deux approches est très faible par rapport aux autres approches.

On note pour le temps t et pour un ensemble connecté cs le nombre d'agents véhicules n_{cs}^t et m_{cs}^t le nombre total de demandes connues par les membres de cs, **MsgCount** pour un seul sous-problème peut être proportionnel à n_{cs}^t pour Selfish, $n_{cs}^t \times m_{cs}^t$ pour ORNInA, $2n_{cs}^t$ pour Dispatching et ${n_{cs}^t}^2$ pour DCOPs. La figure 10.22 à la page 118 illustre la taille moyenne relative des messages (**MsgSize**). En général, la taille des messages d'information sur les demandes (qui est le type de message commun à tous les mécanismes) croît linéairement avec m_{cs}^t, et donc avec la taille du CS. Pour le mécanisme *Dispatching*, nous avons en plus des messages *query* et *response* dont la taille est proportionnelle à $n_{cs}^t \times m_{cs}^t$. Les messages d'enchère et de demande de tirage pour ORNInA ont une taille stable indépendante de la taille du sous-problème, car chacun d'eux est un message biunivoque ne concernant qu'une seule demande à la fois. Il en va de même pour les messages de décisions du DCOP, qui sont de petite taille. Cependant, la densité des messages de décision des DCOPs réduit fortement l'effet de la taille des messages d'information sur les valeurs moyennes de **MsgSize**, ainsi nous pouvons voir les lignes de DSA et MGM-2 dans la Figure 10.22 comme presque constantes.

Le tableau 10.1 présente également la fréquence des reprogrammations de plannings en considérant l'intervalle moyen entre deux cycles de simulation au cours desquels les véhicules mettent à jour leur plannings. Plus cette valeur est élevée, plus les plannings des véhicules sont stables. Dans ces contextes dynamiques, le fait d'avoir des plannings stables pendant une longue période signifie qu'aucune nouvelle demande n'est insérée, ce qui affecte la qualité de service. D'autre part, lorsque les horaires des véhicules changent fréquemment, les véhicules peuvent changer de destination et osciller pendant un certain temps avant d'effectuer un trajet réussi, ce qui peut diminuer la qualité de service. Dans nos scénarios, la coordination fondée sur des DCOPs permet d'obtenir des horaires très stables et de bonne qualité au détriment d'une charge de communication plus importante. Si la stabilité n'est pas une contrainte, mais que la communication est limitée, une approche utilisant des enchères est une très bonne alternative de stratégie d'allocation.

Sommaire

ce chapitre récapitule le contenu de la thèse afin d'en faciliter la lecture pour les francophones. Nous avons proposé un modèle générique pour un problème d'allocation des ressources rencontré dans la gestion de flottes de véhicules autonomes connectés. Notre modèle est bien adapté au domaine de l'ODT, où les flottes répondent en ligne aux demandes des passagers dans des environnements dynamiques. La composante communication de notre modèle prend en charge la transmission directe, par diffusion de messages et transitivité, et repose sur le concept d'ensembles connectés. Nous proposons également un modèle SMA comme support au déploiement de différentes approches pour trouver des solutions et coordonner les véhicules. L'utilisation de ces deux modèles offre une généricité sur les dimensions de la communication et de la coordination. D'une part, la portée limitée de la communication caractérise le problème en affectant le niveau de connectivité et en limitant ainsi les possibilités de centralisation. D'autre part, étant donné qu'il dépend du processus d'attribution, le choix du sous-comportement de planification des véhicules définit le mécanisme de coordination qui affecte le contexte spatio-temporel dynamique des instances du problème.

Appendix D

Conclusion de la thèse

Cette thèse a présenté nos travaux sur le problème d'allocation de ressources dans le domaine du transport à la demande. L'objectif de ce travail est d'explorer différentes méthodes pour résoudre le problème d'allocation de véhicules autonomes aux demandes en ligne de manière décentralisée. Étant donné une flotte de véhicules autonomes déployés dans un réseau urbain, nous cherchons des solutions pour répondre à un grand nombre de demandes de passagers qui surviennent au moment de l'exécution dans différents endroits de la ville. Sans connaissance préalable de la distribution spatiale ou temporelle de ces demandes, les véhicules de la flotte doivent être capables de mettre à jour dynamiquement leurs horaires pour répondre aux demandes nouvellement annoncées. Plusieurs méthodes existent déjà pour résoudre ce type de problème. La solution traditionnelle est fondée sur une unité centrale qui recueille toutes les informations nécessaires pour programmer les véhicules et les répartir ensuite pour répondre aux demandes potentielles. L'un des inconvénients de cette solution traditionnelle est sa charge de calcul et de communication lors de la recherche d'un horaire optimal et à jour. D'autre part, il existe également un certain nombre d'approches décentralisées sur les techniques des systèmes multi-agents dans lesquels un ensemble d'agents autonomes prennent des décisions collectivement et s'adaptent dynamiquement aux changements de leur environnement. L'objectif de ce travail est de modéliser les différents aspects des problèmes de prise de décision et d'optimisation liés à ce problème plus général. Suite à la modélisation de ces problèmes, la question du choix des méthodes de résolution centralisées et décentralisées se pose. Dans ce travail, nous étudions les orientations et comparons les performances des techniques d'optimisation distribuée des contraintes (DCOP), des techniques multi-agents auto-organisées, des approches basées sur le marché et des solutions centralisées de recherche opérationnelle.

Le problème de l'allocation des ressources entre plusieurs entités est une préoccupation centrale en informatique et en économie. Au cours des dernières années, les problèmes d'allocation sont devenus l'un des problèmes d'optimisation les plus étudiés dans la littérature. Dans la partie I, nous avons présenté un aperçu de la littérature scientifique disponible sur les principaux aspects liés au problème en question. Nous fournissons dans le chapitre 1 une vue d'ensemble des différents problèmes de routage de véhicules, des spécifications de Dial-A-Ride Problem (DARP), et de leurs méthodes de résolution existantes dans la littérature de recherche opérationnelle. Dans le chapitre 2, nous passons en revue les efforts déployés dans la littérature de "Multiagent Systems (MAS)" et "Multiagent Resource Allocation (MARA)". Nous concluons de cette revue de littérature que les solutions MARA sont identifiées par les comportements des agents individuels et leurs mécanismes de coordination de sorte que divers modèles de solutions existent. Nous pouvons définir trois aspects pour les caractéristiques de ces solutions qui sont : 1) le niveau d'autonomie de l'agent, 2) le niveau de coopération de l'agent, et 3) le mécanisme d'allocation

de l'agent. Ensuite, nous étudions l'applicabilité de MARA à ODT dans le chapitre 3, d'où nous pouvons déduire le besoin d'une manière unifiée et adaptative de représenter les instances du problème. De plus, les méthodes de solution varient sur de nombreux aspects, de sorte qu'une manière unifiée de représenter et de catégoriser les différentes méthodes de solution est nécessaire pour comparer équitablement et efficacement leurs performances dans différents contextes. À notre connaissance, de telles représentations spécifiques au DARP font défaut dans la littérature de MARA et ODT.

Les réponses aux questions de recherche

Dans cette section, nous résumons brièvement les principales contributions de cette thèse en réponse aux questions de recherche formulées dans l'introduction. Ces contributions ont été présentées lors d'événements scientifiques nationaux et internationaux (ateliers et conférences) évalués par des pairs, et ont donc été publiées sous forme d'articles évalués par des pairs dans les actes, les comptes rendus ou les numéros spéciaux de ces événements, répertoriés sur la page 125. Les principales hypothèses de cette thèse ont été présentées au consortium doctoral de la 30e Conférence internationale conjointe sur l'intelligence artificielle (IJCAI-21) et incluses dans les actes de cette conférence sous la forme d'un article court [P2].

Nous faisons référence à la publication évaluée par les pairs associée à chacune de ces contributions. Dans la partie II, nous avons cherché à combler le vide susmentionné dans la littérature en proposant AV-OLRA, un modèle générique pour le problème d'allocation de ressources rencontré dans la gestion de flottes de véhicules autonomes.

Notre première contribution est de définir un modèle de communication évolutif. Compte tenu de la variété des technologies de communication pouvant être utilisées lors du déploiement de flottes de véhicules autonomes, nous avons dû définir un modèle de communication évolutif pour chaque alternative différente visant à répondre à la **Question de recherche 3**. Nous avons défini ce modèle dans le chapitre 4 (cette contribution est associée à un article présenté lors du 12ème atelier sur l'optimisation et l'apprentissage dans les systèmes multi-agents [P7], inclus comme un un article court dans les actes de la 20ème Conférence Internationale sur les Agents Autonomes et les Systèmes Multi-Agents [P8] et inclus en version étendue dans les actes des Journées Francophones sur les Systèmes Multi-Agents (JFSMA-21) [P9]).

Nous avons commencé par définir la connectivité entre deux composants du système, qui est obtenue par des messages directs dans des portée de communication limitées. Étant un paramètre de déploiement, la portée de communication ajoute une autre dimension de généricité au modèle AV-OLRA. Pour maximiser leur connectivité, si deux véhicules ne sont pas assez proches l'un de l'autre pour communiquer directement, nous leur permettons de communiquer de manière transitive en fonction de l'existence d'un autre véhicule qui est connecté directement ou de manière transitive à ces deux véhicules. Ce modèle de communication permet de réduire la complexité du problème en le divisant en sous-problèmes définis par les membres des ensembles connectés et leur requête connue.

Notre deuxième contribution est de définir les principaux éléments nécessaires à la formalisation du problème. Dans le chapitre 4, nous avons présenté notre modèle générique pour le problème d'allocation de ressources en ligne avec des véhicules autonomes *AV-OLRA*, visant à répondre à **Research Question 1**. Ce modèle définit l'hypothèse du problème (composantes, contraintes) et les indicateurs pour évaluer les différentes stratégies d'allocation (les publications associées sont [P7], [P8], et [P9]). Nous formulons le problème AV-OLRA sous la forme de quatre composantes : un ensemble dynamique de demandes de passagers définissant les ressources ; une flotte de véhicules autonomes définissant l'ensemble des consommateurs ; un graphe orienté connecté représentant le réseau routier urbain et un horizon temporel dans lequel les véhicules doivent répondre aux demandes des passagers. Ce modèle étend un modèle d'état

de l'art pour l'allocation de ressources localisées en ligne (OLRA) en y ajoutant les spécifications de l'ODT et des véhicules autonomes connectés. qui étend un modèle précédent appelé OLRA. AV-OLRA ajoute au modèle précédent les spécifications des ODT et des véhicules autonomes connectés, en l'étendant avec la modélisation des contraintes de communication et de temps des ODT. Ensuite, dans le chapitre chapter:av-olra-mas, nous avons présenté l'architecture MAS pour fournir le modèle AV-OLRA dans lequel les agents peuvent communiquer entre eux via des canaux radio en utilisant des messages peer-to-peer dans des ensembles connectés. Nous avons proposé une exigence de programmation orientée multi-agents pour construire ce modèle générique. Notre hypothèse est que le MAS proposé offre une généricité sur les deux dimensions de communication et de coordination.

Notre troisième contribution consiste à proposer une représentation unifiée des méthodes de solution sous forme de mécanismes de coordination. Avec cette contribution présentée dans [P7], [P8], et [P9] nous avons voulu répondre à **la question de recherche 2**. . Le problème AV-OLRA peut être résolu par différentes méthodes et algorithmes. Ces méthodes peuvent être classées en fonction des propriétés des mécanismes de coordination adoptés. Ainsi, elles peuvent varier d'un dispatching centralisé à des décisions individuelles entièrement décentralisées, et d'un système coopératif à un système compétitif. Dans la partie III, nous décrivons en détail la spécification de chacun de ces différents mécanismes. Dans le chapitre 6, nous donnons un aperçu de la formulation traditionnelle du problème sous forme de programme linéaire et décrivons comment l'adapter au cadre dynamique et considérer les instances du sous-problème AV-OLRA définies par les ensembles connectés. Ensuite, dans le Chapitre Chapitre4, nous explorons la direction de la décentralisation. Nous avons ici trois types de comportements de véhicules : le comportement *Selfish* sans coordination, le comportement *Market-based* de coordination et le comportement *Cooperative* de coordination. Nous avons présenté les spécifications de chacun de ces mécanismes.

Notre quatrième contribution est de combiner le dynamisme et l'amélioration en continu de la solution dans une seule méthode. Pour répondre à la question de recherche 5, nous avons proposé au chapitre 8 un nouveau mécanisme de coordination basé sur le marché dans lequel les véhicules établissent rapidement des horaires réalisables en utilisant des enchères au premier prix pour éviter et gérer les conflits. Puis, à partir de ces horaires réalisables, ils peuvent lancer des enchères de pair à pair pour échanger leurs demandes d'horaires afin d'améliorer la qualité des solutions. Cette approche a été présentée pour la première fois lors du 11ème atelier international sur les agents dans le trafic et le transport (11th International Workshop on Agents in Traffic and Transportation), et lors de deux rencontres françaises (RJCIA-21). [P3] et AFIA/ROADEF-21[1] [P1]) organisé par l'Association Française d'Intelligence Artificielle (AFIA), puis a été publié dans le numéro spécial Agents in Traffic and Transportation (ATT 2020) de la revue AI Communication [P6].

Notre contribution finale est de proposer un cadre expérimental de simulation construit selon les spécifications AV-OLRA. La quatrième partie a été consacrée à la validation expérimentale du modèle et à la comparaison des performances des méthodes de résolution qui suivent une variété de mécanismes de coordination. Cette évaluation était basée sur les critères techniques et qualitatifs que nous avons définis pour évaluer les méthodes de solution visant à répondre à la **Research Question 4**. Dans le chapitre 9, nous avons présenté le génie logiciel qui sous-tend le cadre expérimental et les directives pour mettre en œuvre cinq mécanismes de coordination. Ensuite, dans le chapitre 10, nous présentons les résultats expérimentaux de la comparaison basée sur la simulation entre les mécanismes de coordination implémentés. Ces expériences ont été réalisées sur deux types de scénarios. Le premier est basé sur des données synthétiques générées sur le réseau urbain de Saint-Étienne, tandis que le second est basé sur des données du monde réel extraites des enregistrements de trajets du TLC de la ville de New York. La comparaison est effectuée en termes de qualité de la solution, de charge de communication, de connectivité et de stabilité de la solution. Une partie des résultats de ces expériences a été incluse dans [P6], [P7], et [P8].

[1]Le congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision

Résultats, limites et orientations futures

Les résultats de la simulation montrent que le recours à la DCOP ou aux enchères pour coordonner les décisions décentralisées permet d'obtenir des allocations de qualité raisonnable par rapport à l'allocation optimale one-shot et aux taxis non coordonnés. Les stratégies d'allocation basées sur le DCOP ne changent pas les horaires des véhicules trop fréquemment mais induisent tout de même plus de communication que la stratégie basée sur les enchères.

Une limitation de notre modèle de communication est le phénomène des demandes spatialement obscures. Il s'agit de demandes annoncées loin des véhicules et qui peuvent rester inconnues de tout ensemble connecté pendant un certain temps jusqu'à ce qu'un véhicule passe à proximité de leur source, de sorte qu'elles peuvent ne pas être satisfaites dans les limites de leur fenêtre temporelle. Cependant, dans ce travail, nous avons considéré des scénarios très dynamiques dans les dimensions spatiales et temporelles afin qu'une telle situation ne se produise dans aucun de nos scénarios expérimentaux.

Nous pensons que ce travail mérite d'être approfondi, par exemple en explorant la possibilité de définir des contraintes supplémentaires sur le mouvement des véhicules afin d'obtenir plus de connectivité entre les véhicules ou de s'assurer que chaque source d'émission est située dans l'espace de communication d'au moins un véhicule.

Une autre direction à explorer est la suivante : si les véhicules ont accès à un modèle statistique ou d'apprentissage automatique pour prédire les demandes futures, comment cela pourrait-il affecter la qualité de leurs décisions et quels sont les effets de ces connaissances sur la qualité finale de la solution. Pour améliorer la qualité des décisions de planification des AV, il pourrait être utile d'utiliser des fonctions de priorité plus complexes qui tiennent compte non seulement du contexte actuel du problème, mais aussi d'une certaine anticipation de la façon dont le contexte du problème se présentera dans un avenir proche. Deux questions se posent alors :

- Peut-on faire en sorte que les véhicules prédisent la distribution des demandes dans l'espace et le temps ?
- Comment cette connaissance pourrait-elle affecter la qualité de leurs décisions ?

Dans l'annexe A, nous présentons une proposition de travaux futurs pour répondre à ces questions.

Enfin, nous pensons que le choix entre les différentes méthodes de solution ne peut être considéré comme une décision simple. De plus, il ne s'agit pas seulement de questions techniques. La nécessité de faire correspondre la satisfaction humaine et les décisions contrôlables exige que ces décisions soient transparentes et auto-explicables. Notre vision de l'orientation future est d'appliquer ces résultats à la construction d'un outil analytique entièrement automatisé et auto-explicable qui fonctionne comme un système de recommandation pour les méthodes d'allocation des ressources pour les scénarios ODT. Cet outil potentiel prend en entrée l'ensemble des paramètres du scénario (propriétés du parc de véhicules et modèle de distribution des demandes), la fonction objective de l'utilisateur et ses préférences, en plus du modèle d'environnement (réseau routier et modèle de trafic). Il recommande ensuite des mécanismes d'allocation spécifiques qui correspondent aux objectifs et aux préférences de l'utilisateur. Cette proposition illustrée dans l'annexe B a été présentée au 3rd International Workshop on EXplainable and TRAnsparent AI and Multi-Agent Systems (EXTRAAMAS 2021) et incluse en tant que chapitre de livre dans son post-proceedings [P5]. Nous prévoyons également d'analyser en profondeur la relation entre la stabilité, la complétude et la faisabilité des solutions à l'avenir. Pour ce faire, nous devons mettre en œuvre des approches plus soutenues de différents types et comparer systématiquement les performances, la qualité, la faisabilité, la stabilité et les problèmes techniques pour l'application pratique de ces approches.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *2022LYSEM002*

Alaa DAOUD

Coordination in Connected Autonomous Vehicle fleets :
A Multiagent Resource Allocation Approach to Online On-Demand Transport

Speciality : Computer Science
Keywords : Multiagent Systems, On demand Transport, Resource Allocation, Connected Autonomous Vehicles, Simulation.

Abstract :
The development of autonomous vehicles, capable of peer-to-peer communication, as well as the interest in on-demand solutions (e.g., Uber, Lyft, Heetch), are the primary motivations for this study. More precisely, we are interested here in solving the problem of allocating autonomous vehicles in a decentralized manner. A fleet of autonomous vehicles is deployed to respond to numerous requests from different locations in the city.
Typically, this problem is solved by centralizing the requests in a portal where a fleet manager assigns them to vehicles. This implies that the vehicles have continuous access to the portal (via a cellular network, for example). However, accessing such a global switching infrastructure (for data collection and order delivery) is costly and represents a critical bottleneck. The idea is to use low-cost vehicle-to-vehicle (V2V) communication technologies to allow vehicles to coordinate without a global communication infrastructure.
We propose to model the different aspects of decision and optimization problems related to this more general challenge. Then, the question arises as to the choice between centralized and decentralized solution methods. Methodologically, we explore the directions and compare the performance of distributed constraint optimization techniques (DCOP), self-organized multiagent techniques, market-based approaches, and centralized operations research solutions.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *2022LYSEM002*

Alaa DAOUD

Coordination de flottes de véhicules autonomes connectés :
Approche décentralisée d'allocation de ressources pour le transport à la demande

Spécialité: Informatique

Mots clefs : Systèmes Multiagents, Transport à la Demande, Allocation des Ressources, Véhicules Autonomes Connectés, Simulation.

Résumé :

Le développement de véhicules autonomes, capables de communiquer de pair à pair, ainsi que l'intérêt pour les solutions à la demande (par exemple, Uber, Lyft, Heetch), sont les principales motivations de cette étude. Plus précisément, nous nous intéressons ici à la résolution du problème d'allocation de véhicules autonomes de manière décentralisée. Une flotte de véhicules autonomes est déployée pour répondre à de nombreuses demandes provenant de différents endroits de la ville.
Typiquement, ce problème est résolu en centralisant les demandes dans un portail où un gestionnaire de flotte les répartit aux véhicules. Cela impose que les véhicules aient un accès permanent au portail (via un réseau cellulaire, par exemple). Cependant, l'accès à une telle infrastructure de commutation globale (pour la collecte des données et la livraison des commandes) est coûteux et peut représent un goulot d'étranglement critique. L'idée est d'utiliser des technologies de communication de véhicule à véhicule (V2V) à faible coût pour permettre aux véhicules de se coordonner sans infrastructure de communication globale.
Nous proposons de modéliser les différents aspects des problèmes de décision et d'optimisation liés à ce défi plus général. Ensuite, la question se pose du choix entre les méthodes de solution centralisées et décentralisées. Sur le plan méthodologique, nous explorons les orientations et comparons les performances des techniques d'optimisation des contraintes distribuées (DCOP), des techniques multi-agents auto-organisées, des approches sur marché et des solutions centralisées de recherche opérationnelle.