Machine Learning for the distributed and dynamic management of a fleet of taxis and autonomous shuttles

Tatiana Babicheva

université PARIS-SACLAY

Machine Learning pour la gestion distribuée et dynamique d'une flotte de taxis et navettes autonomes

Machine Learning for the distributed and dynamic management of a fleet of autonomous taxis and shuttles

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Unité de recherche : Université Paris-Saclay, UVSQ, Données et Algorithmes pour une ville intelligente et durable, 78035, Versailles, France.
Référent : Université de Versailles Saint-Quentin-en-Yvelines

Thèse présentée et soutenue à Paris-Saclay, le 10 mars 2021, par

Tatiana Babicheva

Composition du jury

Jakob Puchinger Professeur des universités, CentraleSupélec	Président
Alain Quilliot Professeur des Universités, Université Clermont Auvergne	Rapporteur
Akka Zemmari MCF (HDR), Labri, Université de Bordeaux	Rapporteur
René Mandiau Professeur, LAMIH	Examinateur
Direction de la thèse	
Leila Kloul Maître de Conférences, HDR, Université de Versailles Saint-Quentin-en-Yvelines (UVSQ)	Directrice de thèse
Dominique Barth Professeur, Resp., Université de Versailles Saint-Quentin-en-Yvelines (UVSQ)	Coencadrant
Wilco Burghout PhD, Director of Centre for traffic research, KTH Royal Institute of Technology	Coencadrant
S. M. Hassan Mahdavi PhD, Vedecom, Mobilab	Invité

Thèse de doctorat

NNT: 2021UPASG023

Never send a human to do a machine's job

Agent Smith

In this era of digitalisation and smart technologies, the use of autonomous vehicles as a mode of transport is not a question of 'if' but 'when'

Professor Chen Tsuhan

Neo... nobody has ever done this before.

Trinity

I know. That's why it's going to work.

Neo

Let's see if you ... can do 90

Marty

Of course machines can't think as people do. A machine is different from a person. Hence, they think differently. The interesting question is, just because something thinks differently from you, does that mean it's not thinking?

Alan Turing

I need your clothes, boots and your motorcycle

The Terminator

Abstract

In this thesis we investigate methods to manage electric autonomous taxi urban systems. We consider an online context in which customer demands occur over time, and where vehicles are available for ride-sharing and require electric recharging management. The goal of the investigated system is to satisfy the customers which require the taxi service. The problem belongs to the class of NP hard problems, thus, only heuristic solutions are possible. We propose the heuristics based on problem decomposition which include road network repartition and highlighting of subproblems such as charging management, empty vehicle redistribution and dynamic ride-sharing. The main methods used in these works are based on reinforcement learning.

The first main contribution of the thesis is a set of precalculation tools. For this purpose, the methods of obtaining the exact network structure from GPS data of vehicle movements combined to OSM data are studied. The resulting graph, constructed from both OSM and GPS polyline data, allows obtaining velocity patterns on the arcs, which are the necessary components for aTaxi management. We propose methods of city graph partitioning and we provide a module for the graph covering based on this partitioning.

The second main contribution of the thesis is the development of heuristic methods for local optimisation problems. The set of new methods for empty vehicle redistribution is proposed, both proactive, meaning to take into account both current demand and anticipated future demand, in contrast to reactive methods, which act based on current demand only. One of the main methods uses indexing which is calculated from information on currently waiting passengers, predicted near-future demand and projected arrival of vehicles. The ride-sharing methods aim to identify passengers which can be grouped in one vehicle in order to serve more passengers using smaller fleet. For them, we provide a solution based on city partitioning.

The third main contribution is the reinforcement learning approach. We consider the learning at different levels depending on the targeted granularity of the system. The set of proposed solutions is designed for use in various transport networks. For example, for small networks we propose a station-based RL model, whereas for huge ones, we propose a zone-based RL model, where the agents are subzones of the city obtained by partitioning. The centralised models can show better performances in terms of execution time and stability. We provide also a complete information optimisation in order to analyse the system performances a-posteriori in offline optimisation.

The evaluation of the performances of the proposed methods is provided considering a set of road networks of different natures and sizes. The vehicle-based and station-based reinforcement learning methods, as well as heuristic and exact solutions, were compared on artificial networks of ring and grid structures. The methods for empty vehicle redistribution were compared on test cases in Saclay, France and in Stockholm, Sweden. We then evaluate the performances of zone-based reinforcement learning on a use case of the city of Porto, Portugal. The totality of the results show that proposed methods allow a better scalability and immediate functioning of the electric autonomous taxi system. On all the test cases, the proposed methods provide promising results outperforming the other tested methods and the real data on the taxi system performances in terms of the number of satisfied passengers under

fixed fleet size.

Keywords: autonomous vehicles, taxi management, traffic optimisation, reinforcement learning, empty vehicles redistribution, ride-sharing, electric vehicles.

Résumé

Dans cette thèse, nous étudions les méthodes de gestion des systèmes urbains de taxi autonomes électriques. Nous considérons un contexte en ligne dans lequel les requêtes des clients se produisent au fil du temps, et où les véhicules sont disponibles pour le covoiturage et nécessitent une gestion de la recharge électrique. Le but du système développé est de satisfaire les clients qui ont besoin d'un service de taxi. Le problème appartient à la classe des problèmes NP difficiles, ainsi, seules des solutions heuristiques sont possibles. Nous proposons des heuristiques basées sur la décomposition de problème qui inclue la répartition du réseau routier et la mise en évidence de sous-problèmes tels que la gestion de la charge, la redistribution des véhicules vides et le partage de trajet dynamique. Les principales méthodes utilisées dans ces travaux sont basées sur l'apprentissage par renforcement.

La première contribution principale de la thèse est un ensemble d'outils de précalcul. À cette fin, les méthodes d'obtention de la structure exacte du réseau à partir des données GPS des mouvements de véhicules combinées aux données OSM sont étudiées. Le graphe résultant, construit à partir de données de OSM et GPS, permet d'obtenir des modèles de vitesse sur les arcs, qui sont les composants nécessaires à la gestion des taxis autonomes. Nous proposons des méthodes de partitionnement de graphe de ville et nous fournissons un module de recouvrement de graphe basé sur ce partitionnement.

La deuxième contribution principale de la thèse est le développement de méthodes heuristiques pour des problèmes locaux d'optimisation. L'ensemble des nouvelles méthodes de redistribution des véhicules vides est proposée, à la fois proactives, c'est-à-dire qui prennent en compte la demande actuelle et la demande future anticipée, contrairement aux méthodes réactives, qui agissent uniquement sur la demande actuelle. L'une des principales méthodes utilise l'indexation qui est calculée à partir d'informations sur les passagers actuellement en attente, la demande prévue dans un proche avenir et l'arrivée prévue des véhicules. Les méthodes de partage de trajet visent à identifier les passagers qui peuvent être regroupés dans un seul véhicule afin de servir plus de passagers en utilisant une flotte de taxis plus petite. Pour eux, nous proposons une solution basée sur le partitionnement de ville.

La troisième contribution principale est l'approche d'apprentissage par renforcement. Nous considérons l'apprentissage à différents niveaux en fonction de la granularité ciblée du système. L'ensemble des solutions proposées est conçu pour être utilisé dans différents réseaux de transport. Par exemple, pour les petits réseaux nous proposons un station-agent modèle RL, alors que pour les grands, nous proposons un zone-agent modèle RL, où les agents sont des zones de la ville obtenues par partitionnement. Les modèles centralisés peuvent montrer de meilleures performances en termes de temps d'exécution et de stabilité. Nous fournissons également une optimisation sous information complète afin d'analyser les performances du système a-posteriori en optimisation hors ligne.

L'évaluation des performances des méthodes proposées est fournie en considérant un ensemble de réseaux routiers de natures et de tailles différentes. Les méthodes d'apprentissage par renforcement basées sur véhicule et station, ainsi que des solutions heuristiques et exactes, ont été comparées sur des réseaux artificiels de structures en anneau et en grille. Les méthodes de redistribution des véhicules vides ont été comparées sur des cas de test à Saclay, France et à Stockholm, Suède. Nous évaluons ensuite les performances de l'apprentissage par renforcement par zone sur un cas de test de la ville

de Porto, au Portugal. L'ensemble des résultats montre que les méthodes proposées permettent une meilleure évolutivité et un fonctionnement immédiat du système de taxis autonomes électriques. Sur tous les cas de test, les méthodes proposées fournissent des résultats prometteurs surpassant les autres méthodes testées et les données réelles sur les performances du système de taxi en termes de nombre de passagers satisfaits sous une taille de flotte de taxis fixe.

Mots Clés : véhicules autonomes, gestion des taxis, optimisation du trafic, apprentissage par renforcement, redistribution des véhicules vides, partage de trajet, véhicules électriques.

Acknowledgements

First of all, I would like to thank my UVSQ supervisors Dominique Barth and Leïla Kloul, as well as VEDECOM supervisors Wilco Burghout and Hassan Mahdavi. Their guidance and ability to explain difficult concepts in easy words is the quality, which helped me through all my Ph.D. Their support of me during all these 3 years was inestimable. I am also grateful to my thesis reporters for reviewing my work. And, of course, I thank all the members of comity for accepting to evaluate my work.

The thesis is made in the collaboration with VEDECOM and is a part of the lot 2 of the MOB03 project (multimodal hubs) of VEDECOM that addresses the issue of managing a fleet of autonomous vehicles in the city's infrastructure. Therefore, I would like to thank all my VEDECOM team of MOB03, currently, SYNTONIE, and my colleagues, part of whom become my good friends. I am grateful to Abhishek Jandial for his immense support through all good and bad days. I would also like to express my sincere gratitude to Toussaint Hoche, my table neighbour in both UVSQ and VEDECOM, for his help with my French and administrative problems, as well as for some inspiring conversations. I thank Bofei Chen, Maxime le Guilloux, Najeh Ikram, Maxime Redondin, Vincent Cagnard and Jaâfar Berrada for our discussions in our open-space.

I am grateful to the DAVID team of USVQ for their ambiance. I thank my M1 students who I taught algorithms on graphs, it was a real pleasure to teach you, even if it was in French.

I thank ASFR for the honor they provided to me by awarding me The French-Swedish Prize for young researchers in the PhD student category, as well as for the possibility to visit KTH Stockholm as a visiting researcher. I am grateful to Matej Cebecauer, Erik Jenelius and David Leffler, and other members of KTH Urban Mobility Group who provided me the data of Stockholm and inspired me to new ideas.

Also, I am grateful to all my friends with whom I spent a lot of joyful hours and days and who share my hobbies. I am immense grateful to my friends from ENS Cachan, Veronika Gavrilenko, Ekaterina Katia, Julien Christophe, and Adrien Ditrick, who helped me from my first days in France. I would like to thank Tatiana Shpakova, Vadim Kantorov, Ivan Mitrofanov, Elena Tverdokhlebova, Anna Saraikina, Ivan Novikov, my teammates in the intellectual game "What? Where? When?". I thank my friends all over the world, and, of course, Rina Sheinkin, Maiia Botova, Alexander Kilianov, Alexander Digurov, Liia Sharafutdinova, and the family of Alena, Sergey and Evgeniia Logvinovs for their love and our long conversations. I am grateful for all the team of MIPT Olympiad School, and especially Ilya Mescherin, for their support. I thank my young future colleagues Sonia Soldatova, Sofia Sergeeva, Danila Zhabenko and Alexandre Capdevielle-Fidel for the inspiration they always give to me. I thank Natalia Kilianova for her help with beautiful pictures. And, of course, I am immense grateful to my co-author in the area of history mathematical modelling of traffic flows Ira Ulianova for our historic discussions and her poems.

I would also like to express my sincere gratitude to my parents who opened for me a wonderful world of mathematics and their faith in me. I say thank my brother Dmitry for all his brother care, force, and readiness to come to the rescue at any moment.

Last but not least, I am very grateful to my boyfriend Alexey Novoselov for his love and who was always there in easy and difficult times.

Contents

Chapter 1

Introduction

1.1 Smart cities and smart transport

According to WHO, half of the modern world's population lives in cities. According to the UN forecast, by 2050, about 85 % of the world's population will prefer an urban lifestyle [263]. Citizens will account for about 80 % of world GDP[1]. Rapid urbanization is a challenge for municipalities, which must create the conditions for the expected quality of life, especially in terms of comfort and safety. This is a very difficult task, because the urban infrastructure is already working with extreme load, and with the growth of the population of city agglomerations, there is a threat of its collapse. The concept of **Smart City**, which is one of the attempts to solve this challenge, is being widely discussed in the literature [45].

Smart city means the integration of information and communication technologies for the management of urban infrastructure: transport, education, healthcare, housing systems and security. The goal of a smart city is to improve the quality of life through the introduction of urban information technologies and increase the efficiency of meeting the needs of residents.

An unhealthy transportation system can transform a city from a comfortable living environment into a giant trap. Citizens will not be able to rely on timely medical care, firefighters will not have time to get to the fires on time, and utilities will no longer cope with emergencies. Transport collapse will paralyze the city's economy, and the first companies to be hit will be fleets of public transport, cargo transportation and delivery services. Then the city suffocating in traffic jams and exhaust gases will finally become unsuitable for living.

One of the key tasks of the city authorities is the introduction of technological solutions to create a reserve capacity of transport infrastructure, taking into account the growing load. However, it is important not only to modernize and build new elements of the transport ecosystem, but also to effectively use existing ones.

Transport is changing in accordance with the requirements of the time and the trend towards universal digitalization. With the development of the Internet of Things (IoT) and the advent of smart cities, vehicles will not be able to do without an Internet connection. Connecting a smart vehicle to the network can be beneficial for city authorities. If they collect and use the accumulated data correctly, they will open up new opportunities for working and improving the transport situation in the city. These changes in transportation are referred to as to "**smart mobility**".

Intelligent and environmentally friendly (electric) transport is one of the essential elements of smart mobility. The **ride-sharing** in its basis is a collective trip of several fellow travellers sharing expenses by private car. This movement exists now and has gained popularity due to the availability of mobile technologies. Through online applications, people find those who are with them along the way, for example, to work and even like-minded people to travel. Thus, people get to the right place with great comfort and unload roads and public transport.

[1] Gross Domestic Product

Given the recent rapid development of self-driving (also called **autonomous**) vehicles, organising the ride-sharing will be simplified. Of course, the implementation of the car of the future requires a good software, machine vision, automotive sensors and other advanced systems. For the popularity of such a service among customers, it is important that it turns out to be more comfortable, faster and more economical than standard transport. In addition, autopilots have hopes to reduce accident mortality and improve transport safety.

1.2 aTaxi management issues

The objective of this thesis is concentrated around electric aTaxis[2] development. More precisely, the goal is to design the reservation strategies allowing for a dynamic sharing of these vehicles by several customers. The basic idea of these strategies is to identify the customers whose requests can be grouped and assigned to the same vehicle while guaranteeing a certain quality of service.

In order of the formalisation of the research problem, the question is "*what do we need to be able to evaluate the system?*" This leads to the following first research issue. How do we define both the quality of service *to be guaranteed to the customer in electrical aTaxi system* and *the quality of service for the system itself.*

This issue leads to the need of customer profiles identification by recognising the patterns of temporal and spatial distributions of the runs made by autonomous vehicles from data collected in big cities (New York, Stockholm, ..). Customer demand can vary greatly from day to day and week to week. In addition, commuting can be very directional at specific times. These factors lead to an imbalance between the number of available vehicles in different regions. Thus, efficient service requires a redistribution of empty vehicles to match the supply of vehicles to passenger demand. Analysis of displacement data can improve statistical prediction of future displacement demands.

In order to provide the relevant mathematical and computational model, we need to formalise the environment of the taxi system. Thus, there is a need to *represent and to generate consistent road network graph.*

Autonomous vehicle reservation strategies rely on grouping and machine learning techniques with distributed agents (autonomous vehicles, potentially capable of exchanging information) and a centralised system with knowledge of the environment. Depending on the required quality of service and locations of the vehicles, the system will propose to process the received customers requests according to its own utility. The strategies of all these actors will be refined through the use of automatic learning and prediction methods. This autonomous taxi system faces two major management problems: "*Which vehicles to assign to which passengers?*", and "*How to redistribute empty vehicles?*". Both problems are NP-hard. Therefore, classical optimisation methods are limited to heuristic or "greedy" solutions, especially for networks with big size and real structure. In addition, the electrical nature of the taxis leads to the question "*How to model and manage the charging of the vehicles?*".

Evaluation of the gap between the user equilibrium and system optimum is another challenge. In order to deal with it, we should answer the question "*Is it necessary to model the reservation management system at different levels?*", and, if yes, "*Which levels should be investigated?*". The same gap reveals the optimisation problem in the case of complete information and expands it to the gap between global and local optimums. This leads to the question "*How to evaluate the performance of the system in the approach of "price of anarchy"*".

All these research questions should be addressed based on realistic scenarios in the context cited above.

[2] autonomous taxi

1.3 Problem definition

This thesis addresses the issue of managing a fleet of autonomous vehicles in the city's infrastructure. We consider an area such as city or city region with the stations of the transport system. The city offers a service of autonomous taxis and shuttles moving on a road infrastructure shared with other types of vehicles (public transport, vehicles with human driver). A set of parking spaces equipped with electric charging stations, accessible not only to autonomous taxis, but potentially to any electric vehicle, is spread throughout the region. The traffic is known or approximated by the global management system that interacts with all aTaxis.

This system will allow customers to book an autonomous vehicle, either in advance or on the fly (for a trip as soon as possible). The proposed route may be shared with other passengers or may require the client to walk a certain distance to get the taxi. The main objective of a transport operator offering aTaxi service is to efficiently use the available vehicles. The operator seeks to maximise effectively the quality of service while minimising total operational costs, which include reducing energy consumption, vehicle maintenance and other expenses.

The **problem setting** of the thesis is the following. Given the city or city region with all the structure and demand patterns, how to manage effectively the electric aTaxi system? Thus, given the certain level of quality of the service, how to determine the best strategies for the vehicles and the system to reach the user equilibrium and system optimum?

1.4 Contributions and thesis outline

The results presented in this thesis have diverse nature but are combined into a single framework. This thesis consists of 7 chapters.

Chapter 2 is dedicated to state-of-the-art. The focus is made on searching answers or approaches that exist in literature to our research questions, such as empty vehicle redistribution, ride-sharing, taxi charging and taxi system management. The state-of-the-art reveals the patterns in taxi systems and provides a justification of the chosen methodology. The main accent is made on machine learning approaches to different transportation modelling problems, including the taxi system simulation.

In Chapter 3 we describe the model of electrical aTaxi system and formulate the real-time aTaxi management problem. This problem highlights optimisation subproblems such as charging management, empty vehicle redistribution and dynamic ride-sharing.

In Chapters 4 and 5 we present our management optimisation methods of the aTaxi system. These methods are based on the reinforcement learning and are provided at different levels, called micro (where agents are the vehicles), meso (where agents are the city zones, obtained by clustering), and macro (where the shared learning is between agents). These methods use different additional modules based on heuristics, which aim to provide improvements in the solution performance.

Chapter 6 is devoted to the evaluation scenarii description, including a new method of relevant weighted city road graph construction. This method is based on OpenStreetMap (OSM) network data and real vehicle movement GPS data sets together. We obtain a graph that combines strong points of both data types: the exact geographical structure of OSM data and the real movements of vehicles of GPS data. Then we describe clients and vehicles characteristics and generation patterns. The last sections of this chapter describe the networks and the scenarii to investigate further.

Chapter 7 is dedicated to the evaluation of proposed methods on scenarii proposed in Chapter 6. We provide the characteristics of the convergence and performances of proposed methods on five different networks. The methods based on clustering showed faster convergence and less computational complexity. We illustrate our findings using simulations on real and synthetic data, and classical benchmarks with many observations.

In Chapter 8 we provide the thesis conclusion where we resume our contributions and explore some research perspectives. Part of the results provided in this thesis was published and presented in several journals, conferences and seminars.

Publications during the thesis

Published journal papers

1. Tatiana Babicheva, Matej Cebecauer, Dominique Barth, Wilco Burghout, and Leïla Kloul. Empty Vehicle Redistribution with Time Windows in Autonomous Taxi Systems. ACM/IMS Transactions on Data Science, January 2021, Article No.: 5 https://doi.org/10.1145/3416915, 22 pages.

2. Tatiana Babicheva and Wilco Burghout (2019). Empty vehicle redistribution in autonomous taxi services. EURO Journal on Transportation and Logistics, 8(5), 745-767.

3. Tatiana Babicheva, Wilco Burghout, Ingmar Andreasson, and Nadege Faul (2018). The matching problem of empty vehicle redistribution in autonomous taxi systems. Procedia computer science, 130, 119-125.

4. Tatiana Babicheva, Wilco Burghout, Ingmar Andreasson, and Nadege Faul (2018). Empty vehicle redistribution and fleet size in autonomous taxi systems. IET Intelligent Transport Systems, 13(4), 677-682.

5. Matej Cebecauer, Wilco Burghout, Erik Jenelius, Tatiana Babicheva and David Leffler. Integrating Demand Responsive Services into Public Transport Disruption Management. IEEE Open Journal of Intelligent Transportation Systems, doi: 10.1109/OJITS.2021.3057221.

6. Tatiana Babicheva, Wilco Burghout, Ingmar Andreasson, Nadege Faul. 'The matching problem of empty vehicle redistribution in autonomous taxi systems (extended version). Special Issue in the International Journal of Traffic and Transportation Management (JTTM). doi: 10.5383/JTTM.01.01.001

International conferences

1. Tatiana Babicheva and Irina Ulianova.The history of mathematical modelling of traffic flows in the 1920-1970s. Kazan digital week - 2020, September 2020

2. Matej Cebecauer, Erik Jenelius, Tatiana Babicheva, David Leffler, and Wilco Burghout. Public Transport Disruption Management by Collaboration with Demand Responsive Services. Transportation Research Board 2020 Annual Meeting, Washington, D.C., January 12–16 2020

3. Tatiana Babicheva and Irina Ulianova. The history of mathematical modelling of traffic flows in the 1920-1970s. International Conference of Transport, Traffic and Mobility, Paris, Sorbonne, October 2019

4. Tatiana Babicheva, Wilco Burghout, Dominique Barth, and Leïla Kloul. Construction and analysis of a weighted city road graph using OpenStreetMap and statistical GPS data. ITS World Congress, Eindhoven, June 2019

5. Tatiana Babicheva, Wilco Burghout, Ingmar Andreasson, and Nadège Faul. Empty vehicle redistribution and fleet-size in autonomous taxi systems. 25th ITS World Congress, Copenhagen, September 2018

Seminars and workshop talks

1. Tatiana Babicheva. Machine learning for the distributed and dynamic management of a fleet of taxis and autonomous shuttles. Journée de doctorants of VEDECOM, 10 October 2019

2. Tatiana Babicheva. Methods for the distributed and dynamic management of a fleet of autonomous taxis and shuttles. Interdisciplinary Research Program on Urban Mobility, September-October 2019, Université Paris-Saclay.

3. Tatiana Babicheva. Distributed and dynamic management of a fleet of autonomous taxis. The French-Swedish workshop on Smart cities and mobility, 11-12 June 2019

4. Tatiana Babicheva. Ridesharing and empty vehicle management in autonomous taxi systems. The Conference Scientifique Annuelle SMIV (Smart Mobility and Intelligent Vehicle), 27 November 2018

5. Tatiana Babicheva. Machine learning for the distributed and dynamic management of a fleet of taxis and autonomous shuttles. Journée de doctorants of VEDECOM, 12 October 2018

Scientific visit and collaboration

- Visit to KTH Stockholm granted by ASFR, Mai 2019.
- Moderator of the section "ITS development in the World changing context", Kazan digital week - 2020, September 2020
- Participation in development of VIPSIM program in VEDECOM. Implementation of algorithms for the optimization of a fleet of autonomous taxis (redistribution of empty vehicles, ride-sharing).

Chapter 2

State-of-the-art

2.1 Introduction

Personal Rapid Transit (PRT) [108], also referred to as "podcars", is an on-demand transportation system that uses a set of small driverless (and often electric) vehicles to transport people on demand from point to point (or station to station). Research within PRT [17] has traditionally assumed guideways (elevated or not) exclusively used by the PRT vehicles. Due to the construction cost of such guideways, as well as the rigidity of the available routes they provide, the world has seen only a handful of implementations of PRT services, e.g. Morgantown PRT [121] and Heathrow [169].

Given the recent rapid development of self-driving (also called autonomous) vehicles, the need for guideways is disappearing, as autonomous taxis will be able to use existing roadways, and thus provide door to door public transport services. In addition, the current trend towards "Transport as a service" (TAAS) [116] or "Mobility as a Service" (MAAS) [292] where cars (and bicycles, etc.) are shared using increasingly convenient on-line platforms, revolutionises the market for transportation of people and provides new ways to match planned and unplanned travel demands with transport services. Thus one can consider PRT and autonomous taxi services to be equivalent. An essential prerequisite for any such service is the efficient operation of the vehicle fleet, maximising the service to the passengers while minimising the cost to the operator(s).

Mathematical modelling, including computer simulation, can answer a significant number of questions arising when optimising the traffic in PRT and autonomous taxi services. One of the ways to achieve this is to develop adequate mathematical models of the processes of interaction of vehicles with such transport infrastructure elements as the road network and traffic management systems. As a rule, models of this type belong to the class of imitation or micromodels, which combines cellular automata models, multiagent models, etc. The currently popular micromodelling algorithms require a significant number of computational operations. From this point of view, the development of macroscopic models and algorithms for optimising transport processes, which, on the one hand, easily adapt to high-performance computing tools, and, on the other hand, would have reserves to reduce computational complexity, becomes urgent. To bring reserves to life it is required to solve yet unsolved modelling and optimisation problems of large dimension with more accessible computing facilities, as well as for the evaluation of certain processes in real time. It is especially important to find a solution of the optimisation problem which reduces the gap between user equilibrium and system optimum, and using well planned shared autonomous vehicles is the way to this.

In order to build a relevant model with all the characteristics required, a deep study of the methods already used in the literature is required.

To define the quality of service to be guaranteed to the customers, it is necessary to study the mathematical, economical and sociological models of passengers. Is it necessary to minimise the average passenger waiting time or the maximum waiting time? Can a passenger leave the system after obtaining the answer from the taxi service that he will need to wait more time than he would like? Are passengers

ready to walk some distance in order to leave earlier? Is a passenger ready to share his trip with other passengers, and if so, under what conditions?

To identify customer profiles by identifying the patterns of temporal and spatial distributions of the runs made by autonomous vehicles from big cities data, there is a need to examine what type of data the existing datasets contain, and whether this is enough for the problem setting. How to define customer profiles? Is the use of machine learning techniques already required at this stage, and, if so, which techniques for analysing such data are currently in use?

To model the reservation management system at several levels, or, more precisely, at the level of individual vehicles and at the level of a fully centralised system, we should determine the computational complexity of each of the subtasks. We should also understand whether exact solutions are possible, or heuristic solutions are required. Moreover, it is necessary to study which heuristic solutions are being proposed now, as well as for which size of the transport networks and with which characteristics they are applicable.

One important component in any taxi management system is the way in which vehicles are assigned to passengers and how they are redistributed in the network to provide an optimal service. In both PRT and autonomous taxi systems the service is demand-responsive, most passengers request immediate service and only part of trips are booked ahead of time. Perfect information about future requests is therefore not available, but statistical information about future requests may be predicted from historical data. In addition, the passenger demand is often asymmetric across the network and over time, causing imbalances in vehicle supply and passenger demand in the system. Therefore, it is necessary to redistribute empty vehicles based on demand and supply attributes to rebalance the system. *To propose relocation strategies for empty vehicles to provide better service,* it is required to study what techniques are currently used, and also to think out strategies based on classical heuristic methods, as well as on reinforcement learning.

The complex and interdisciplinary nature of the thesis objectives determines several different areas of the literature research.

Subsection 2.2.1 describes preferences of passengers. This part is core for a more accurate definition of the model's objective functions and for the understanding of expected quality of service nature. A small Subsection 2.2.2 dedicated to electrical vehicles is needful to model the charging strategies of vehicles. Subsection 2.2.3 is devoted to the empty vehicle redistribution methods. In the model without empty movement, the taxi will not depart to meet the passenger, that will not converge to an optimal solution. Even in classical problems with human drivers, the empty vehicle sending is essential. The autonomous vehicles have less empty ride costs, and this leads to the fact that the research of the potential improvements is necessary. In Subsection 2.2.4 we consider the existing methods of ride-sharing in taxi systems. Systems with implemented ride-sharing can serve more passengers, and, especially in congested traffic condition, can help to obtain the optimum with fewer vehicles required. Subsection 2.2.5 is focused around the machine learning approaches to different transportation modelling problems, including the taxi system simulation. The review of the methods most commonly used nowadays is provided. Section 2.3 discusses the directions and methods chosen in the thesis. Finally, Section 2.4 provides the conclusions to the state-of-the-art.

2.2 Existing approaches in transportation problems

2.2.1 Customer preferences in transportation systems

People in developed countries spend many hours of their life traveling to work and to home, and waiting taxis and trains. Thus it would be profitable to lessen this time. The time of consumer becomes one of goods to buy. The consumer follows the microeconomics rules to choose the transportation method, summarizing (or choosing another type of utility estimation) travel and waiting times to total cost of consuming good. This fact is often not to take into account, but the consumer would prefer to spend as less time in transportation as it is possible.

In the most economic models the goods are separated in categories of leisure, which add utility when consumed, and work, which is the source of income to purchase the goods to increase utility. There is trade-of between leisure and work: the more time is spent for work the less it is spent for leisure. For each individual it matters how to reassign the available time between leisure and work. In transport investments, the concept of "time value" (value of time, VOT) is widely used. Despite the fact that each action has its own value for an individual, when trying to model a group's behaviour to determine estimates, difficulties arise: how to get the sources of individual distributions? They can be obtained as a result of population surveys (method of declared preferences), as well as through econometric models (method of revealed preferences, RP). None of these methods is perfect and the results of their use diverge from research to research.

Becker's model [28] is classical and the time in his model is the main goal to optimize. He uses the VOT that is equal to the individual wage rate. Lisco [179] considers the value of transit time to be equal to the half of the individual wage rate. For Watters [280], this value is of 30 to 60 percents of the wage rate.

Utility function, developed by Johnson [135] uses work time, so the leisure time became different from other authors. Oort [206] proposes to include transportation time in his model where the working time increases when the travel time decreases. The travel time also has negative effect to utility function (he considers the travel time as some kind of unpleasant activity). Mohring et al. [199] introduce a passenger disutility as a linear function of time spent walking to and from bus stops and waiting there for service, and time spent actually travelling. For de Serpa [77], the utility function depends on activities and on goods. Evans [84] considers a utility function as function with arguments which are the times using in activities. Diaz [133] observes the fact that time now is considered to be as the main utility component and all activities lead to changes in the utility value. Thaithatkul et al. [260] describe ride-sharing system considering linear utility function depending on waiting time and travel time. Lenoir and Laplace [171] consider the price of the travelling time and the activities and conditions that can affect it. Schakenbos et al. [234] study the transfers in public transport, and show that the disutility during an interchange depends on travel time, the waiting time and the headway.

The results obtained using the RP data do not allow accurate calculation of general trends in people's preferences due to the low accuracy of the data collected and the strong variation. Using Stated Choice (SC) data to determine the cost of time has been the standard since the mid-1980s. Since this type of survey allows to process more data and also to obtain preference patterns not only in waiting or travel times, but also in the type of used transport. Since the mid-1990s, algorithms have appeared that allow to correct and detect correlations in responses from each person, which reduced some of the advantages of using the SC data method, but still did not take away the palm from this method.

One of the limitations of the SC analysis [70] consists of the value of small time savings for common travels and the policy in eleven different countries. The authors considered utility difference linear on time and money. The paper addresses the issue of how small time savings are handled in the appraisal of transport proposals. The authors provide critical discussion of the treatment of small time changes that have been estimated in SC studies and conclude that there is a strong case for reconsidering the use of RP data.

When studies showing the VOT as being primarily based on an economic analysis of utility functions, *the value of travel time changes (VTTC)* (the monetary measure of the value that people attach to changes in travel time), uses at the same moment the psychological aspect of choice preferences. VTTC is a key factor in evaluating and comparing different transportation projects. Saving time on the road often constitutes a large part of the economic benefits of a project. Thus, overcoming a given constraint is crucial for cost-benefit analysis [277]. Several countries are conducting national studies to evaluate official VTTC, which can be used to assess and prioritise transport development projects. In design of the most of SC experiments the citizens choose between possible scenarii evaluating a time-cost trade-off [243]. Such methods were used, for example, in the Netherlands [262], Norway [85], the UK [187], Denmark [97], and Sweden [41].

In order for studies to be comparable, the relevant normalisation is to be provided. The corresponding values are provided by such authorities as "Commissariat général du plan" [40] in France or the department for transport in the UK. Algers et al. [7] describe the first Swedish national VOT study, which followed the methods used in the UK and Netherlands studies.

Work in the mid-1990s confirmed and quantified non-linearities in VOT and revealed the following non-linearities:

- loss of time is more important for the subject than the acquisition;
- the amount of time or cost savings relative to the total amount of time or money spent on a trip;
- small times have diminished VTTC;
- there may be preferences for current time depart.

Subsequent studies with a more complex analysis of the responses about SC in general confirmed and corroborated these findings. However, the real question is whether or not the data obtained are correlated with reality. If they correlate, then how to take them into account? If not, what experiments and how to conduct, so that they correlate the results with reality? How to take these effects into account when evaluating a study?

Hultkranz and Mortazavi [130] use the Swedish data to reveal non-linearities of VOT. Börjesson and Eliasson [41] based on the data collected through two surveys carried out in 2007 and 2008 clearly show the non-linearity of the VTTS. Also, authors show that the commuting/school trips have around 30% higher VTTS than other private trips. Yang et al. [298] consider the Cobb–Douglas type production function of taxi–customer meetings considering the walking time and the taxi waiting time.

In the past decade, models based with error defined on the log time and log cost difference scale, and with value of time distributed randomly in the population are appeared. They are more difficult to understand and implement in simulation, but for now they most fully explain all the effects. Multinomial logit models and more recent mixed logit models are also widely used to evaluate VTTC [41]. There is no clarity in literature yet in defining and classifying the main modelling approaches used for the data sets of the type described above. In general, the methods of parametric and non-parametric estimation are highlighted [205]. More informative parametric models allow the VTTC to vary depending on the covariates, which is significant in connection with the detected non-linearities. Basing on the national VTTC studies, two basic approaches can be highlighted. These parametric approaches, named Random Utility (RU) and Random Valuation (RV) differ in the random component belonging. In the RU it refers to the difference between the utilities of the trip options. In the RV it belongs to the difference between the proposed assessment threshold and the actual value of the cost of travel time, implied in changes in time and proposed cost. From the deterministic point of view, these approaches can be considered as equivalent and both can be deduced from standard microeconomic theory. The significance of the differences in them has not yet been clarified in modern literature and the approach is usually chosen empirically [97], [41], [129] or from the preferences of the authors. Classics in the discrete choice models Ben-Akiva and Bierlaire [32] develop the RU theory using the multinomial probability unit model. Authors consider, in addition to travel time, the other attributes such as path length, travel cost, transit specific, etc. Ojeda-Cabral, Batley, and Hess [205] focus on official national binary time/money trade-offs VTTC studies. Authors consider RU and RV approaches to analyse the VTTC at Danish and UK datasets with the conclusion that RU approach is likely to contain heteroscedastic error terms that need correction.

2.2.2 Electrical vehicle charging and discharging

Currently, three main types of electrical powered vehicles can be highlighted.

- Battery Electric Vehicles (BEV) do not use any power sources than electrical power, obtained from batteries based onboard.
- Hybrid Electric Vehicles (HEV) contain both electrical batteries and traditional gas engine (internal combustion engine, ICE).

- Plug-in Hybrid-Electric Vehicle (pHEV) is in many aspects similar to HEV except the fact the main power source is electricity. ICE is used only for charge batteries, but power outlets can be used. pHEV can work without using fuel at all if there is an power outlet in reach.

In this work, the Battery Electric Vehicles are considered.

EV charging performs by electric vehicle service equipment (EVSE). There are three main levels classified by charge rate and they uses respectively 120V for standard house outlets, and 240V and 480V for the special charging stations.

Total charging time greatly depends of many factors: batteries capacity, amount of required charge power, needed charging speed (if applicable). No many vehicles use charge cycles from 0% to 100%. Overnight charging usually recharge the batteries from 60-70% to 100%. "Dinner" charges usually compensate up to 1/3 of battery capacity.

Fetene et al. [94] analyse the driving range and investigate the factors affecting the energy consumption rate of fully-battery electric vehicles under real-world driving patterns accounting for weather condition, drivers' characteristics, and road characteristics. Energy consumption rate was found to be equal to 0.168 kWh/km during the summer and 0.225 kWh/km during winter.

Hiermann et al. [122] consider linear discharging based on the travelled distance.

Schneider, Stenger and Goeke [235] study an E-VRP[1] with time windows and charging stations (E-VRPTW) that minimizes the total distance travelled by a homogenous EV fleet. The model considers the battery charging time to depend on the remaining battery level upon arriving at the charging station. The battery consumption in the article is a function of travel distance (travel speeds on arcs are assumed to be constant).

Erdogan and Miller-Hooks [83], Desaulniers et al. [76], Felipe et al. [93] present an E-VRP with multiple charging technologies and partial recharges. Again, the energy consumption in all the works mentioned considered to be a linear function of travelled distance.

Based on the literature research, in the optimisation problems with electrical vehicles the classical approach is to consider the linear with the distance discharging and linear with the time charging.

2.2.3 Empty vehicle redistribution

One of important issues mentioned before is the empty vehicle redistribution (relocation, reposition) problem (EVR problem) in PRT systems which can be formulated as a dynamic version of the Vehicle Routing Problem (VRP) which aims to find the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers [78]. In the VRP with time windows, the passengers need to be picked up at stations within defined time windows and/or delivered at their destination within given time windows. Both problems can be reduced to the Traveling Salesman Problem (TSP), and are therefore NP hard. As a result, most approaches in literature use heuristics rather than exact methods, to find acceptable solutions within reasonable time.

The methods for EVR can be divided into two main groups corresponding to two different problems. First group, reactive redistribution strategies, in which the call for a vehicle to be made at the moment of passenger arrival, are methods of reaction to demand. In such algorithms, an empty vehicle is to be sent in interests of particular passengers (the one with maximal waiting time, the one closest to the vehicle, etc). Second group, proactive redistribution strategies (sending empty vehicles in anticipation of predicted demand in the near future) is the empty vehicle redistribution in interests of all the system.

In some PRT systems [121], two types of operations with another notation are used. In the demand mode of operation, there are no heuristics or knowledge of future demand, EVR is based only on the current demand, that makes the problem close to the dynamic problem. In the schedule mode of operation, there is information or strong heuristics about the future passenger demand, that makes the problem close to the static deterministic problem.

Kek, Cheu, and Chor [153] implement two staff-based relocation techniques called "shortest time" and "inventory balancing". Shortest time relocation is based on calling an empty vehicle from the

[1] Electric vehicle routing problem

nearest station (with the shortest time). Inventory balancing relocation calls vehicles to stations with fewer vehicles than necessary, assigning vehicles from stations with over-supply (surplus). The model is based on a "virtual station status", which is the real number of vehicles in the station at this moment, plus weighed total number of vehicles scheduled for return to this station (for this time interval), minus the number of vehicles which are not available for use (because they have been reserved). When the "virtual station status" violates the threshold values for the station, a relocation request is issued.

Song and Earl [246] consider the problem of determining optimal EVR and fleet-sizing policies for a two-depot service system with Poisson arrivals and independent exponential random times needed for repositioning. Authors go on to minimize the sum of costs incurred by vehicle maintenance, EVR, and the leasing of vehicles.

Specialist in dynamic programming Powell [216, 217] uses operator profit maximising as objective function.

Bruglieri, Colorniand, and Lue [46] propose the movement model for one-way electric car sharing EVR problem. It was believed that electric cars are able to pass a fixed distance on one battery which charges and discharges linearly (from time and traveled distance respectively). The objective function in this staff-based reallocation model represents the total number of satisfied requests.

Boyaci, Zografos, and Geroliminis [42] describe a model for the planning of electrical one-way non-floating reservation-based car-sharing systems, for both pick-up and drop-off. The objective function in this model is maximising the net revenue for the operator. The model has a high computational complexity and does not take into account any information on future arrivals.

Fagnant, Kockelman et al. [87, 88, 155] focus on shared autonomous vehicles' mileage and environmental implications. Their EVR method is based on the nearest available vehicle and uses 2-dimension cellular automaton.

Hanna et al. [119] consider car-sharing with autonomous vehicles as a multi-agent system. This study is partially based on a work of Fagnant and Kockelman. The authors propose the "SCRAM" algorithm based on Huhn's Hungarian algorithm [159] to minimise the maximum edge cost in a bipartite graph.

Anderson [12] proposes EVR based on operational experience, and a flow of empty vehicles can be started in anticipation of passenger arrivals. This model is based on centralised control EVR between nearest stations.

Fan, Machemehl, and Lownes [89] develop the dynamic vehicle allocation problem for car-sharing (DVAPC) solutions. The objective of proposed model is to maximise the total expected profits over a planning horizon of N days under the condition of independent random future demands with a known distribution.

Bell and Wong [31] propose a reactive nearest neighbour (NN) algorithm for EVR. When a new passenger arrives, the proposed heuristic immediately selects a vehicle to serve him or her and adds the passenger to the list of ordered trips for that vehicle. It is assumed that the dispatcher knows the trip request probability for each OD[2] pair and the average headway between requests.

Daszczuk et al. [72] propose a disaggregate redistribution algorithm based on inter-station data exchange, with the possibility of limiting the distance between communicating stations.

Lees-Miller [169] focuses on passenger waiting times in PRT systems. The SV (Sampling and Voting) [170] heuristic is based on the Dynamic VRP method. Each subtask that appears in the model is solved by a nearest neighbour algorithm and the best solution is chosen using a voting system. In [168], the author also uses the "longest-waiting passenger first" heuristic, comparing it with Bell and Wong's nearest neighbours algorithm. It is shown that although the Bell and Wong method may saturate at higher demand intensity, the longest-waiting passenger algorithm may decrease queues and shorten waiting at lower demand.

Febbraro, Sacco, and Saeednia [92] develop car-sharing simulator with user-provided methods of EVR. In this method, the provider pays not for staff, but for users, and the total price for users decreases, possibly leading to the demand increase.

[2]Origin-Destination

Kornhauser from Princeton Autonomous Vehicle Engineering [157, 158] focuses mostly on aTaxi systems. Zhu and Kornhauser [307] propose methods of city partitioning (called pixelisation). The passenger's order will be served with a vehicle from the closest pixel. Under supervision of Kornhauser, Douglas [80] focuses on a model of shared aTaxi system for New Jersey. The EVR and fleet-sizing problems for an aTaxi system tackle both deterministic and stochastic demands. The objective is to find the policy that minimises the expectation of the sum of travel costs of sending all dispatched empty vehicles to their destinations and a "lateness penalty" for all "departure units" that were scheduled to dispatch before the decision time, but have not yet departed. In this model, lateness penalties are assigned to "departure groups" proportional to time but independent of the number of passengers in each "departure unit". Gladstone, under supervision of Kornhauser, also proposes a solution based on the pixelisation of network. Every 30 minutes the model performs check if it is optimal for each empty vehicle to be repositioned. If so, the vehicle would move one pixel toward the optimal direction. Thus, this method is not relevant to the significant one-dimension flow during the rush hour. The ride-sharing strategy provides multiple stops with limits on detours, but the picking up at intermediate stops is not considered. Thus, the maximal vehicle occupancy for Manhattan, New York, was always less than 1.7.

Fatnassi et al. [90, 91] introduce the "battery restricted" PRT problem with empty-run minimising as the objective function [12]. The strategy assigns vehicles to passenger requests in real time without knowing the expectancy of next request. The authors propose four reactive strategies for vehicle movements as follows.

- First-come, first-served (FCFS). The available vehicle serves the passenger with maximum waiting time disregarding distance.
- Nearest neighbour (NN) from the vehicle perspective. Each vehicle serves the passenger from the closest station.
- Wait, then go (WG). After serving a passenger request, a vehicle will wait in its station for more passengers for a given time. If no trip request happens within this time period, the vehicle will serve its nearest passenger request based on the NN strategy.
- Fixed number of vehicles in station (FNVS). In every station there should be a fixed number of waiting vehicles. As soon as a station has a deficit of waiting vehicles, the nearest waiting vehicle from a station with vehicle surplus will be sent to this station.

The algorithms were evaluated on a number of randomly generated PRT networks. The evaluation is done based on empty versus effective (loaded) vehicle movements, as well as average passenger waiting times. The authors find that the WG outperforms the other algorithms in every aspect, although the Nearest Neighbour shows very similar results.

Van der Heijden et al. [266] use a repository of requests (including future requests if any), empty vehicles, and occupied vehicles expected freeing. They evaluate the following algorithms based on the empty vehicle mileage produced: a) FCFS algorithm; b) a look-ahead algorithm anticipating known near-future demand; c) a hierarchical model that combines global and local empty vehicle managers; d) integral planning algorithm improving on c) by considering all requests and empty vehicle trips together.

Beaujon and Turnquist [27] identify interactions between fleet-sizing and levels of fleet utilisation throughout the time period over which EVR decisions are made. This model assumes static demand and has a profit maximisation as an objective.

Andréasson [16, 15, 14, 13] implements and evaluates reactive and proactive redistribution strategies for large networks with ride-sharing. The studies are mostly focused on high passenger loads close to the maximum capacity of the vehicle fleet-size and network. The objectives are to minimise the passenger waiting times (average and maximum) and the number of unserved passengers at the end of the simulated period. At frequent intervals all empty vehicles in stations or on links (called or sent) are reallocated (exchanging destinations between vehicles) with priority given to serving the longest waiting passenger.

Maciejewski et al. [186] evaluate a reactive nearest-idle-vehicle (nearest neighbour) algorithm as well as a proactive demand-supply balancing algorithm. Their findings indicate that for low passenger

demand the algorithms show similar results, but for high demand scenarios the balancing algorithm outperforms the nearest-idle-vehicle algorithm.

2.2.4 Ride-sharing research

The impact of vehicles on the environment is one of hot issues in modern world. The introduction of ride-sharing can not only expect to diminish traffic jams and therefore to decrease the passengers' travel times, but also can decrease total vehicle run times and extra stops on the way. The most closely related routing problem to ride-sharing is the PDP[3] that are known to be NP-hard. Ride-sharing can be modeled as a one-to-one PDP. The E-VRP-RS (Electrical vehicle routing problem with ride-sharing) can be also considered as an extension of the vehicle routing problem with multiple depots (MDVRP), which requires different vehicles leaving from and returning to different locations. Since the VRPs are proved to be NP-hard, the PDPs are known to be NP-hard, thus, new methods of heuristic solutions have to be developed.

Furuhata et al. [105] propose a classification according to the different aspects of the existing ride-sharing systems, which includes the following 6 classes according to the various criteria identified.

Dynamic real-time ride-sharing provides the matching between drivers and passengers in real time (en-route). The matching agencies provide strategies for routing, scheduling, and pricing.

Carpooling provides more predicted matchings, such as regular trips with similar OD-pairs.

Long-distance ride-match provides more flexible departure time and OD pair, however, comparable with the trip length.

One-shot ride-match is a hybrid of carpooling and long-distance ride-match.

Bulletin-board is a board-based ride-sharing, where ride-sharing conditions are determined by negotiation among the participants.

Flexible carpooling is a semi-organized service, i.e., usually does not use any matching agency. In such a service, the spontaneous ride-sharing is formed based on a first-come first-service basis at the meeting points.

Authors identify three major challenges for ride-sharing services: design of attractive mechanisms (pricing, incentives), proper ride arrangement (user profile information, multi-hop rides), and building of trust among travelers in online systems.

Amey, Attanucci and Mishalani [11] focus on different challenges occurred with the dynamic ride-sharing appearance. There are economic challenges; social/behavioral challenges; institutional challenges; technological challenges.

The psychological and environmental challenges are widely discussed in the literature. Merat, Madigan and Nordhoff [195] focus on social and psychological factors of the ride-sharing for automated vehicles and consider users' motivation to use the AV and the ride-sharing depending of various factors. Epperson [82] examines people attracting methods including price policies to ride-sharing systems in order to reach a critical mass. Levofsky and Greenberg [173] provide a research of the ride-sharing systems existed 15 years ago with the reasons of failure or success. Deakin, Frick and Shively [74] report that most people prefer anonymous locations (for example, parking lots, major intersections) rather than giving away their precise start and destination points. Chan and Shaheen [58] focus on historical and psychological points of view about the ride-sharing in mostly the north America. Murray and Steele [202] propose the system and method for facilitating and encouraging ride-sharing basing on awards. Bistaffa et al. [39] concentrate on so-called social ride-sharing with the following approach. Given the desired starting point and destination and the time windows of the commuters, they can form groups that share cars to lower associated transportation costs (i.e., travel time and fuel), while considering the constraints imposed by the social network. Kamar and Horvitz [145] consider challenges with coordination among self-interested people aimed at minimizing the cost of transportation and the impact of travel on the environment. The two ride-sharing optimisations in the article are: (1) generating ride-share plans for groups of agents and (2) clustering agents into ride-share

[3] Pick-up and Delivery Problem

groups. A ride-share plan is defined as a single trip or a chain of trips near the origins and destinations of the other trips. Jacobson and King [132] focus on the fuel consumption model in the ride-sharing systems. The monetary value of fuel savings to the passengers are at least as large as the value of time. Caulfield [56] examines the patterns of ride-sharing in Dublin, and estimates the environmental benefits of the ride-sharing both in terms of reductions of emissions and the total vehicle kilometrage traveled. Andréasson [15] gives an overview of the operational issues created by ride-sharing, including the implications for passenger safety and security. Adler, Miculescu and Karaman [2] consider a class of problems, applicable to vehicle platooning and passenger ride-sharing in autonomous systems. Sending more vehicles at once allows for more energy-efficient transportation, but to do it consistently the station must delay some vehicles to wait for others to arrive.

The technological challenges, especially ride-sharing matching methods, are the most connected with the subject of the current work. In modern literature, a set of articles, patents and methods can be highlighted.

Tom LeWinson and Sharon LeWinson in patent [174] show the application for ride-sharing and describe the ride-sharing logic for specific cases. The first method includes searching for other users whose origin is within a specified distance from the seeker's travel route. The second method includes searching for other users whose destination is within a specified distance from the seeker's travel route. The third method includes combination of the first and the second methods.

Maciejewski and Nagel [186] use multiple pick-up and drop-off locations. However, the exhaustive nature of the proposed simulation makes it limited in scale. The base of the MATSim[4] model used in the article is in the DVRP[5] Optimizer and it uses a mimetic algorithm, consisting of an evolutionary algorithm and a local search procedure for solving the DVRP.

Kornhauser et al. [156] use pixelisation of the city. The matchings are based on the waiting in the departure points in order to reassembly more possible passengers in one vehicle.

Fagnant and Kockelman [88] focus on the fleet-size of the SAV[6] services. The basis for sharing are the following conditions

(1) Current passengers' trip duration increases for not more than 20%.
(2) Current passengers' remaining trip time increases for not more than 40%.
(3) New traveler's total trip time increases by not more than maximum between 20% of total trip without ride-sharing and 3 minutes.
(4) New travelers will be picked up at least within the next 5 minutes.
(5) Total planned trip time to serve all passengers is not larger than remaining time to serve the current trips + time to serve the new trip + 1 minute drop-off time, if not pooled.

The search process prioritises the trips with the earliest final drop-off time.

Levin et al. [172] consider the possibility of dynamic ride-sharing. Following the principle of FCFS, they give precedence to the longest-waiting traveler. However, they allow other passengers to enter the SAV if they are traveling to the same or close destination. In general, VRP can admit solutions in which a SAV picks up several passengers before dropping any off. The heuristic in this study does not do it due to complexity.

Kleiner, Nebel and Ziparo [154] propose the DRS[7] based on the artificial intelligence methods, using sealed-bid second price auctioning scheme.

Wang, Dessouky and Ordonez [275] consider a pickup and delivery problem in which a driver can use HOV[8] lanes and/or receives toll savings when traveling with a certain number of passengers. Authors provide an integer programming formulation and present a Tabu search heuristic.

Cordeau and Laporte [66] propose a Tabu search algorithm for the DVRP. The authors describe a procedure for neighborhood evaluation that adjusts the visit time of the vertices on the routes so as

[4] The multi-agent transport simulation
[5] Dynamic VRP
[6] Shared autonomous vehicle
[7] Dynamic ride-sharing
[8] High Occupancy Vehicle

to minimize route duration and ride times. Berbeglia, Cordeau and Laporte [33] present basic, lazy and eager scheduling which represents the dynamic version of the algorithm proposed in [66]. They solve static dial-a-ride problem minimizing the ride time violation without increasing the time window violation of any node in the route.

Campbell and Savelsbergh [52] provide a comprehensive review on insertion heuristics for VRP with complicating constraints. The same authors in [53] focus on inventory-routing problem and on optimisation-based approach for its solution. This model does not allow multiply deliveries.

Alonso-Mora et al. [10] propose ride-sharing algorithm and implement the rebalancing for the data of Manhattan taxi system. The idea is based on maximizing the amount of requests served while minimizing the cost. Authors propose an anytime optimal algorithm for batch assignment of a set of requests to a set of vehicles, which minimizes cost function, satisfies set of constraints, and allows for multiple passengers per vehicle. The ignored requests in the model are largely penalized. The method proposed by authors not only diminish the fleet-size, but also can make the waiting times smaller.

Ghosh, Page and De Roure [107] propose an application of a machine learning convex optimisation framework called Network Lasso that uses the ADMM[9] optimisation for learning and dynamic prediction. It is an application of a robust and scalable unified optimisation framework within the ride-sharing case study. This model implements the clustering of the network users.

Lees-Miller, Hammersley and Davenport [167] propose single origin and single or multiple destinations ride-sharing model based on queuing theory.

Lin et al. [178] attempt to solve the route optimisation for the ride-sharing taxi problem with simulated annealing (SA) algorithm [35].

Santos and Xavier [232] consider the dynamic ride-sharing problem from the point of view of the combinatorial optimisation. The passenger constraints imposed in their problem formulation consist on departure time, arrival time, and price of the trip. The objective function of the Dynamic Dial-a-Ride Problem with Money as an Incentive (DARP-M) maximizes the number of attended requests and minimizes the total value paid by all passengers. The proposed method is based on the allocation of set of static problems which are solved by a greedy randomized adaptive search procedure (GRASP).

Goel, Kulik and Ramamohanarao [111] present an approach to station-to-station ride-sharing with analysis of the number of stations required from the point of view of the 1-coverage problem.

Ma, Zheng and Wolfson [184, 185] propose a DRS framework for the GPS-equipped taxi cabs. This framework is based on the schedule insertion satisfying capacity, time window, and monetary constraints.

Baldacci, Maniezzo and Mingozzi [25] adopt a Lagrangian column generation method in order to compute the optimal way of assigning commuters to cars while minimizing unmatched participants. Ghoseiri, Haghani and Hamedi [106] propose a method similar to [25] but also try to fulfill individual preferences such as age, gender, smoke, and pet restrictions.

Winter and Nittel [285] consider a setting which uses short-range wireless communication devices, showing that limiting the information dissemination between agents provides a benefit in terms of computation requirements, while it does not significantly impact the solution quality. This paper investigates ad-hoc shared-ride trip planning in transportation networks.

Xing et al. [294] consider a highly dynamic multiagent-based simulation for a ride-sharing system where drivers and riders are matched en-route with an analysis provided by FIPA[10]-compliant multiagent-based simulation system PlaSMA[11]. In this model it is possible that there are no appropriate drivers that can take the passenger the whole way from start to destination, so the passenger agent adapts its search strategy and begins to seek compound transfers with a single change of vehicle.

Thaithatkul et al. [260] aim to investigate day-to-day dynamics and characteristics of passenger matching problem in smart ride-sharing system.

Agatz et al. [5, 3, 4] examine DRS by seeking for the minimum of total (system-wide) vehicle miles

[9] Alternate Direction Method of Multipliers
[10] Foundation for Intelligent Physical Agents.
[11] Program name

travelled (VMT). The detour flexibility of 20 minutes is shown to dramatically improve ride-share matches.

Stiglic et al. [247] focus on single-driver, single-rider ride-sharing system. The objective is to quantify the impact of three different types of participant flexibility that are relevant in the context of DRS, i.e., matching flexibility, detour flexibility, and scheduling flexibility, and to investigate the required level of additional flexibility that is required to improve the effectiveness of a ride-share system.

Wang, Agatz and Erera [274] investigate how large the price of anarchy might be in a practical ride-sharing setting and develop optimisation approaches that use stability considerations when generating ride-share matches. Authors focus on systems where at most one pickup and delivery can take place during the trip and where riders do not transfer from one driver's vehicle to another.

Based on the state-of-the art, we divide the ride-sharing on the categories based on the trajectory shapes, similar to [105].

1. Identical ride-sharing. Both the origin and destination of current trip a and new passenger b are identical.

Figure 2-1 – Identical ride-sharing.

2. Inclusive ride-sharing. Both the origin B and destination B' of new passenger b are on the way of an original route $A \rightarrow A'$ of a vehicle a.

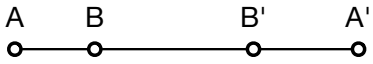


Figure 2-2 – Inclusive ride-sharing.

3. Partial ride-sharing. The new passenger b has the origin B which is on the original route $A \rightarrow A'$ of a vehicle, but the destination B' of the new passenger is not on the way. Ride-sharing is only a part of passenger b's trip.

Figure 2-3 – Partial ride-sharing.

4. Detour ride-sharing. Not pick-up nor drop-off locations of passenger are not on the way of an original route of a vehicle.

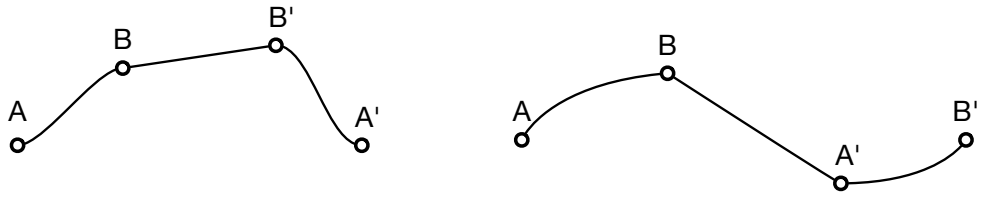


Figure 2-4 – Detour ride-sharing.

Thus, taking a detour, ride-sharing route covers both the origin and destination locations.

The detour ride-sharing covers the biggest amount of potential shared trips. Based on this considerations as well as on the detour flexibility concept, this category is a basis for the work.

2.2.5 Machine learning techniques in transportation

In this section we are providing the classification of the machine learning techniques using in different problems and subproblems of the traffic optimisation.

Generally, machine learning field is divided by researchers and practicians to two main sections: supervised learning and unsupervised learning, but the mixed methods are continuously appearing in different applications.

Supervised learning can be characterised by an existence of training data, thus, generally speaking, it is changing of a function (learning) that maps an input data set to an output data set using example input-output pairs as patterns (training examples). In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. In the framework of the current work, the first idea of the implementation of the supervised learning is teaching the aTaxi system how to divide the client calls to groups "take it" or "don't take it".

Unsupervised machine learning [124] is the problem of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm.

Reinforcement learning in most of the literature is considered as supervised learning. However, the supervisor in this case is represented by environment or its model. Some rules of reinforcement learning can base on implicit supervisors, for example, in the case of artificial neural environment, these rules can base on joint action of formal neurons. Thus, reinforcement learning can be also part of unsupervised learning. This is one of reasons why reinforcement learning in some sources is recognised as one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

Supervised learning techniques

Supervised learning techniques used in transportation modelling are the following.

Ensemble learning To obtain better results ensemble learning [207] is using multiple learning algorithms. It allows to get better results comparing with using learning algorithms alone. Ensemble learning techniques can be divided by the following categories, however, their combinations are also widely used.

Bootstrap aggregating (bagging) [43] is based on uniform weight voting in the ensemble. The variability in the method consists of training on random subsets. The training set of each base classifier is sampled by randomly selecting a subset with replacement. The bagging classification is based on the majority voting scheme.

Bayes optimal classifier (BOC) places all the hypotheses into the hypothesis space. If that hypothesis were true the vote show likelihood that the training dataset would be sampled from a system. The prior probability of the hypothesis is used as multiplicand to get new votes.

Bayesian model averaging (BMA) seeks to approximate the BOC. It combined sampled hypotheses typically using some Monte Carlo method such as MCMC and then using Bayes' law.

Bayesian model combination (BMC) is an algorithmic correction to BMA. This method samples from the space of possible ensembles, where model weights are drawn randomly from a Dirichlet distribution with uniform parameters. This modification overcomes the tendency of BMA to converge toward giving all of the weight to a single model.

Boosting [151] is a meta-algorithm for primarily reducing bias. In the boosting method classifiers are created sequentially so that the next base classifier assigns more weight to samples in which the previous classifier has mistaken. Boosting focuses on the question of Kearns and Valiant: "Can a set of weak learners create a single strong learner?" [152].

One of the most popular methods in transportation is ADABoost. AdaBoost is sensitive to noise in data and emissions [102]. However, it is less prone to retraining compared with other algorithms of machine learning. In [103], authors provide the comparison of the AdaBoost (linear programming) and support vector machines (quadratic programming), showing that different norms can result in very different margins.

Friedman [104] proposes the gradient boosting strategy TreeBoost in application to several popular loss criteria: least-squares, least absolute deviation, Huber, and logistic binomial log-likelihood. Qian et al. [220] provide boosting Gaussian conditional random field (boosting-GCRF) model to accurately forecast the short-term taxi demand distribution using historical time-series demand over the study area. Roor et al. [228] use AdaBoost with Decision Stumps (one-level decision trees) as classifiers to the spatio-temporal next location prediction. For the prediction itself, the Partial Matching (PPM) on a Variable-order Markov Model (VOMM) is used.

Chen et al. [60] propose ensemble learning approach for understanding ride-splitting behavior of passengers of ride-sourcing companies who provide prearranged and on-demand transportation services. This paper employs the boosting ensemble by growing individual decision trees sequentially and then assembling these trees to produce a powerful classification model.

Bucket of models is introduced in [44]. It combines models for each problem and ensemble them to obtain the best model.

Stacking [288] is used to learn how the base classifier make errors, and the second level classifier tries to overcome the errors.

Decision tree learning It is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Discrete tree models are commonly referred to as classification trees, and continuous ones are called regression trees.

The decision trees are widely used in different aspects of vehicle movement and demand predictions. Liu et al. [180] consider Vehicular Ad Hoc Networks (VANETs), integrating the capabilities of new generation Peer-to-Peer (P2P) wireless networks with vehicles. Using the vehicle trip history, authors develop a heuristic and context-dependent induction method based on decision trees to predict vehicle moving trajectories. Zhang and Haghani [304] employ a gradient boosting regression tree method to model and analyze freeway travel time improving the prediction accuracy and model interpretability. Saadi et al. [231] use boosted decision trees for the spatio-temporal estimation of the demand.

Random forests or random decision forests correct for decision trees' habit of overfitting to their training set ny the construction of a multitude of decision trees.

Linear classifier It provides a classification basing on the linear combination of element characteristics, called feature values (typically represented as so-called "feature vector"). There are two classes of methods for determining the parameters of a linear classifier called generative and discriminative models. Methods of the first class model conditional density functions. Examples of such algorithms include the following:

Linear Discriminant Analysis (or Fisher's linear discriminant) (LDA) assumes Gaussian conditional density models.

Naive Bayes classifier with multinomial or multivariate Bernoulli event models.

The second class of methods attempts to maximize the quality of the output on a training set. Examples of discriminative training of linear classifiers include the following:

Logistic regression is a maximum likelihood estimation which assumes the possibility to generate a data set by binomial model.

Perceptron attempts to fix all errors encountered in the training set.

Support vector machine (SVM) aims to maximize the margin between the decision hyperplane and the examples in the training set. This method can be also considered as a special case of Tikhonov

regularisation. Montazery and Wilson [200] use an SVM inspired method to the problem of ride-matching.

Auto-regressive methods Auto-Normal model or Conditional Auto-Regressive (CAR) model is a model of continuous Markov random field and it described by a conditional probability density function. It especially suitable for modelling spatial phenomena closely related to a specific local context.

Besag [36] proposes methods of statistical analysis for lattice scheme, using the conditional probability approach.

The CAR model is also used by Mariella and Tarantino [189], who propose a new The model of Spatial Temporal Conditional Auto-Regression (STCAR), which allows you to process the spatial dependence between the sites, as well as the temporal dependence between the implementations, in the presence of measurements recorded in each spatial location for the time interval. This model is based in part on the Generalized Multivariate Conditional Autoregression (GMCAR) model of Gene et al. [134]. The STCAR model directly determines the joint distribution of a sequence of Markov random fields [68] through conditional and marginal distributions, using information obtained from the time evolution of the phenomenon.

Min and Wynter [197] adopt a multivariate spatial-temporal autoregressive (MSTAR) model to account for transient behaviour on the traffic network and develop a method for traffic prediction at a fine granularity and over multiple time periods.

Inferential theory of learning (ITL) It was introduced by Michalski [196] and it views learning as a search through a knowledge space, guided by learning goals. The search operators are instantiations of certain generic types of knowledge change, called knowledge transmutations (or knowledge transforms). Transmutations change various aspects of knowledge; some of them generate new knowledge, others only manipulate knowledge.

Wojtusiak et al. [287] investigate theoretical and practical aspects of learning by autonomous agents within stochastic agent-based simulation. It describes the agent's learning processes using the ITL, and relates them to the simulation problems. This extension of ITL describes how artificially generated data from simulations can provide the experience necessary for agents' training. Through their interaction with stochastic simulation environments, intelligent agents gather experience. That experience, combined with agents' background knowledge and the information exchanged between agents, can be used for learning. In multi-agent systems, the objective of learning is to improve agents' performance on a specific problem.

Unsupervised learning techniques

Unsupervised learning techniques in classical literature are classified to 4 main categories: clustering, anomaly detection, neural networks and approaches for learning latent variable models. The mixed methods of unsupervised machine learning are also popular. Here are several methods that can be applied to the subproblems appearing when modelling autonomous taxi system.

Neural networks Neural Networks [267] (ANNs) systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. In different sources, the neural network approach can be considered as part of supervised learning, as part of unsupervised learning or even as not part of machine learning at all.

Xu et al. [295] concern with the modelling of the complex demand-supply relationship in urban taxi services. A training algorithm in this article is based on the back-propagation algorithm. The NNs were trained with the annual survey data.

De Brébisson et al. [73] focus on predicting the destination of a taxi using Recurrent Neural Network, Bidirectional Recurrent Neural Network and Memory Networks approaches.

Karlaftis and Vlahogianni [150] propose state of the art concerned statistical vs neural networks. Authors show, that in transportation research, NNs have been mainly used as data analytic methods because of their ability to work with massive amounts of multi-dimensional data, their modeling flexibility, their learning and generalisation ability, their adaptability and their – generally – good predictive ability. Another comparison of NN with non-NN technique is provided by Cheu et al. [61], who compare neural networks and support vector machine approaches for a multiple-station not-dynamical shared-use vehicle system.

Clustering Clustering is used in a lot of different approaches. For the center search, the problem is similar to the NP-hard facility location problem [120], when it is necessary to determine the coordinates of one or more goods, where each of the users minimizes his or her costs (user costs can be expressed as a function of distance from the good, the travel time, price of the ticket, ...) for the minimisation of the function (mostly, the sum) for all the users [22]. Facility location problem or location analysis problem is a field of operation research and computational geometry that analyses objects' placement, optimal from standpoint of minimizing transportation costs with some constraints in effect. Facility location problems typically represent long-term strategic planning issues in supply chain design and they are affected by lower level tactical and operational decisions. The mathematical structure of the location problem is determined by the configuration of feasible region and quality assessment of the accommodation. As a result, there are many various location problems, and technical literature is replete with methods of their solution.

Location problems set two main criteria for evaluating location quality: minimisation of the maximal distance and minimisation of the total sum of distances. Thus, there are two corresponding tasks. The problem of searching for a location point chosen with maximal distance minimisation criteria is called k-centre problem [100, 191, 113, 131, 160], and searching for a point chosen with total distance minimisation criteria is median problem.

More generally, there are five key discrete location models: the set covering model, the maximal covering model, the k-median model, the uncapacitated fixed charge location problem and the k-centre problem [71].

Facility location problem without capacities is classical and set [120] necessity to determine the coordinates of one or more goods, where each of the users minimizes its costs (distances from the good, the travel times, etc.) for minimisation of the function (mostly, the sum) all the users [22].

In the big data the k-median problem [282, 303] arises, which minimizes the sizes of k clusters containing all the points of the consumers' set. The problem is NP-hard on non-trivial graphs [148, 149]. The theoretical limit of approximation ratio is 1.463 [117]. The currently best algorithm has the approximation ratio 1.488 [175]. The most cited and used algorithm is Weiszfeld's algorithm [281]. Unfortunately it may not converge or may do so very slowly. There have been proposed a lot of modifications to Weiszfeld's algorithm that in general give non-asymptotic runtime guarantees [65]. In non-metric formulation there exists a solution with the approximation ratio $O(\log N)$ [125].

Heuristic solutions are: genetic algorithms [126], simulated annealing [112], tabu search [109, 110], node partitioning, node insertion, node substitution, and various hybrids [20, 127, 306, 30]. Good overview on the development of heuristics for solving the k-median problem is described in [198]. Meta-heuristics [255] can be applied to obtain good result from many optimisation problems. The k-means problem [188] minimizes the sum of squared distances instead of the sum of the distances. The transport problems use the sum of the distances as the norm. k-medoids algorithm is characterized by the point of location of goods belonging to the set of consumer points.

Facility location problem with capacities is of special interest. For instance, let there is a need to construct a few park-and-ride stations, placing it according to a certain map, or chart a timetable for a certain bus/train line. No one would build a parking lot which not be used only to 10 percent of its capacity. Again, extremal constraints may represent impossibility of using more than maximal number of parking places.

A generalisation of the classic problem of determining the location of an object occurs when we

associate a capacity with each tool that measures the maximum number of clients that the tool [244, 269] can serve. Further variations of this capacitated facility location (CFL) arise when we limit the number of objects that can be opened in a specific location. Thus, in k-CFL we can open a maximum of k objects anywhere.

The paper [194] describes k-medoid CFL problem on graph reducing the problem to the MIP-problem[12] with constraints. Obtained NP-hard problem then is solved using branch-and-bound method with cut-off heuristics.

All before mentioned papers deal with "no choice" variant of the problem, when each client can be served only by nearest facility (or choose only one good). If there is a choice for each of consumers for what of the goods to choose, the problem becomes much more complicated and now includes an underlying subproblem: which good will be chosen by the consumer if nearest good is full. For example, such a problem arises when searching a parking or charging place. The paper [249] describes algorithm "Search maximal utility regions" to solve this underlying problem. The complexity of given algorithm greatly depends of factor $|C|/|F|$, where C is set of consumers and F is set of facilities (goods). When this factor is small the problem becomes practically unsolvable. Another optimisation methods for the capacitated facility location-routing problem on graphs are mentioned in [208].

For the uniform capacities one of the newest algorithms is obtained by [115]. The algorithm solves uniform capacitated knapsack median problem violating the budget by an additional cost of f_{max} in the facility opening cost and a loss of factor $(3+\varepsilon)$ in capacity. Constant factor $(O(1/\varepsilon))$ approximation for uniform capacitated k facility location at a loss of $(2+\varepsilon)$ in capacity and using at most $k+1$ facilities. The result of $O(1/\varepsilon))$ for uniform capacitated facility location at a loss of $(1+\varepsilon)$ capacity is promising for this class of problem.

Genetics methods are used to solve relatively large dimension optimisation problems. They are based on concepts from the nature, when some population changes each step, so it belongs to class of population based heuristics. Introduced in 1997 by Price and Storn [248], the Differential Evolution (DE) method became widely popular and is used to solve multiparameter optimisation problems such as batch fermentation process, non-linear chemical process, finding of coefficients of non-linear equations and much more.

This method uses a simple evolutionary strategy and it is proved to be one of fastest methods for solving continuous optimisation problems for finding the objective function's global minimum, especially for unimodal functions [183]. It has a few of tuning parameters and requires a small computer code. The authors of DE in 2005 [218] proposed ten modifications to improve algorithm's behaviour in worst cases.

This method can be easily parallelised for use both on a single computer and on their set, including the cluster. The book [268] describes many neural network methods, and DE amongst them. In [203] the modification of DE using non-linear simplex method by Nelder and Mead [204] to generate good initial population is discussed. The modification improves result by finding the solution with better value in the same time or faster. The papers [210, 211] compare DE method with another stochastic method of Swarm of Particles to optimise objective functions with up to 100 parameters. DE shows best result in most of cases.

Reinforcement learning

Reinforcement learning (RL) [136, 253] is widely used method for trajectory prediction and autonomous vehicle modelling. It is an area of machine learning inspired by behaviourist psychology, where agents act in an environment to maximize their subjective profit. The problem, because of its generality, related also to another areas such as game theory, control theory, operations research, information theory, simulation-based optimisation, multi-agent systems, swarm intelligence, statistics, and genetic algorithms. In the operations research and control literature, reinforcement learning is called approximate dynamic programming, or neuro-dynamic programming.

[12] Mathematical Integer Programming

In another words, the reinforcement learning is a type of machine learning in which the agent learns to act in the environment, performing actions and thereby developing intuition, after which it observes the results of its actions. This is one of the most widely used and rapidly developing technologies in the field of machine learning. The paradigm was proposed by Richard Sutton and Andrew Barto, Richard's research supervisor, who helped him to prepare his PhD [253] in the 1980s and was then archaic. Afterwards, Richard believed that it had a great future, and it would eventually receive recognition. Unsupervised learning and deep learning are more designed to identify patterns in available data. In RL, on the other hand, such experience is gained through trial and error; the system gradually finds the right options or global optimum. A serious additional advantage of RL is that in this case, it is not necessary to provide an extensive set of training data, as in learning with a supervisor. A few small fragments will suffice.

In [136] the RL is defined as "the problem faced by an agent that must learn behaviour through trial-and-error interactions with a dynamic environment". This paper discusses central issues of RL from a computer science perspective, where both historical basis of the field and a broad selection of current work are summarized. Some major features of RL such as trial-and-error search and delayed reward are reported in book [253] containing RL techniques as well as historical and bibliographical remarks.

In single-agent RL, the environment is mostly described by a Markov decision process. The agent maximises, by each action the expected discounted return. One of possible methods consists on by computing an optimal action-value function Q (Q-function). Q-learning, introduced by Watkins in 1989 [278], is a method used in artificial intelligence in agent approach. On the basis of the remuneration received from the environment, the agent forms a utility function Q, which subsequently gives him the opportunity to choose a strategy of behaviour, and to take into account the experience of the previous interaction with the environment. One of the advantages of Q-learning is that it is able to compare the expected usefulness of available activities without creating an environmental model.

Busoniu, Babuska, and De Schutter [50] discuss the MARL[13] techniques for fully cooperative, fully competitive, and mixed (neither cooperative nor competitive) tasks. The described MARL technique is based on Q-learning. The complexity of MARL is exponential in the number of agents: each agent is, therefore, faced with a moving-target learning problem: the best policy changes as the other agents' policies change.

Lin et al. [176] offers a contextual multi-agent reinforcement learning system including context algorithms for deep Q learning and contextual multi-agent actor-critic, achieve clear coordination between a large number of agents adapted to different contexts. The authors point out such serious technical problems when modelling fleet management using DRL[14]:

- Feasibility of problem setting. The sequence of actions from the policy is evaluated solely by the reward signal [21], therefore, the RL structure is based on reward, and it requires definitions of the agent, reward and action space.
  The constraints such as number of available vehicles in the problem setting make it a sort of problem of high-dimensional exact-constrain satisfaction policy optimisation, that is not computationally tractable in DRL. Even for a very small-scale problem the computational complexity can be very high. [213].
- Large-scale Agents. One alternative approach is to consider each available vehicle as an agent, which allows you to use the DRL setting for multiple agents. But this approach lead to curses-of-dimension problem due to large size of the space of action. Most of the existing studies [96, 182, 258] allow you to coordinate the work of only a small set of agents due to high computational costs.
- Coordination and Contextual Dependence of Action Space. The action space changes dynamically over time, as agents move to different locations, and the number of possible actions depends on the geographical context of the location.

[13] Multi agent reinforcement learning
[14] Deep reinforcement learning

Tan [259] considers the situation with the fixed model with the same number of independent as well as cooperative agents in order to answer the questions "Given the same number of reinforcement learning agents, will corporative agents outperform independent agents who do not communicate during learning?" and "What is the price for such cooperation?". Based on one-step Q-learning algorithm, authors show that sharing learned policies or episodes among agents speeds up learning at the cost of communication; for joint tasks, agents engaging in partnership can significantly outperform independent agents although they may learn slowly at the beginning.

Foerster et al. [96] offers a multi-agent actor-critic method called counter-actual multi-agent (COMA) policy gradients. COMA uses centralised critic to evaluate Q -functions and decentralised actors to optimise agents' policies. This method is based on independent Q-leaning, but an actor-critic is used instead of Q-learning. This approach is called the Independent Critical Actor (IAC) [54].

Dai et al. [69] focus on the problem of longitudinal vehicle control and present the controller architecture consisting of QEN (Q-estimator network) and FIS (fuzzy inference system by Takagi-Sugeno [297]) to solve continuing reinforcement-learning problems. QEN plays the role of approximation or prediction of the function of the value of the optimal action associated with various input and output control states. QEN is used to evaluate the value function of optimisation, and FIS is used to obtain control output based on the value function of the intended action provided by QEN.

Tamagawaa et al. [256] present a methodology for evaluating city logistics measures considering the behaviour of several stakeholders associated with urban freight transport using a multi-agent model that consists of a learning model and a model for vehicle routing and scheduling problem with time window-forecasted (VRP-TW-F). Authors describe a model that uses Genetic Algorithms (GA) to solve the problem. Authors select two techniques of Q-learning [279] and Monte Carlo method. They construct the learning model using both techniques, and compare the performance of them.

Sastry et al. [233] consider a team of learning automata involved in an N-person game with different payoffs to different players, in general. Here, a Linear Reward-Inaction (LRI) learning algorithm for the game based on a decentralized team of Learning Automata (LA) [261] is presented. This learning algorithm is completely decentralized and can be thought of as a parallel distributed network similar to neural network models. Authors prove that this algorithm will converge to the Nash equilibria.

Unsal et al. [265] propose a Linear Reward Penalty (LRP) method for the intelligent controller for vehicle path planning in unmodelled traffic. Alipour [8] proposes a LRP based method for the capacitated vehicle routing problem.

Chorus, Arentze, and Timmermans [63] propose the random regret-minimisation model of travel choice that is an alternative to random utility-maximisation.

Bertsekas [34] describes methods of approximate solution of stochastic optimal control problems using a neurodynamic programming/reinforcement learning methodology. The focus of the article is on rollout policies, which are obtained by a single policy iteration starting from some known base policy and using some form of exact or approximate policy improvement. Secomandi [239, 238] considers the single vehicle routing problem with stochastic demand. The method is based on employing a rollout algorithm to route the vehicle through the customers concurrently with service. However, the cardinality of the state space in the problem is extremely large.

Proper and Tadepalli [219] present ML approaches to mitigate each of 3 common curses of dimensionality in real-world domains (explosions in state and action spaces, and high stochasticity). To deal with high stochasticity, authors introduce a new algorithm called ASH-learning, which is an afterstate version of H-Learning. The H-learning is a learning which optimizes undiscounted average reward [19].

Sniezynski et al. in [245] describe one of possible methods to improve the fact that the RL suffers from inefficiency when the number of potential solutions to be searched is large. Authors apply rule induction in multi-agent systems. Authors used AQ21 [286] method from IQ class of machine learning programs that induce attributional rules from data and prior knowledge. Experimental results obtained in the article indicate that in this domain the method performs better than traditional Q-learning, as indicated by statistical comparison.

The mimetism learning [296] allows cooperation between agents in order to share the behaviours

already enhanced by the reinforcement learning. The mimetism learning works on the assumption that an agent imitates the behaviours of the most efficient agents. At each cycle, the agent examines the fitness value (objective function) of the best solution found by each other agent of the coalition.

Meignan et al. [192] propose a population based metaheuristic (Coalition-Based Metaheuristic, CBM), adopting the metaphor of social autonomous agents. Agents cooperate and self-adapt in order to collectively solve a given optimisation problem. In CBM the agent's behaviour is guided by a decision process for the operators' choice which is dynamically adapted during the search using reinforcement learning and mimetism learning between agents. The approach described is applied to the VRP to emphasize the performance of learning and cooperation mechanisms.

Wiering [284] proposes a car-based value functions for the problem of traffic light optimisation (traffic control) problem in the network. He uses average waiting time V function (without knowing the traffic light decision) for a car at fixed moment and place until reaching its destination address. The global state is decomposed into local states based on each individual vehicle. The transition function maps one vehicle's location at a given time step to its location at the next time step. As a result, the number of states grows linearly in the number of points and can scale to much larger networks. Furthermore, the transition function combines experience data gathered in different locations, rather than having to learn separate mappings for each location. Thus, the Q-functions in the system depend on V-function.

Kuyer et al. [163] extend the reinforcement learning approach to traffic control by using cooperative learning and explicit coordination among agents. Authors use model-based reinforcement learning, in which the transition and reward functions are estimated from experience and then are used to find a policy via planning methods like dynamic programming. A full transition function have to map the location of every vehicle in the system at one timestep to the location of every vehicle at the next timestep. It is clear that a exact function would use too large for practical usage argument space, but learning a model is nonetheless possible if a vehicle-based representation (as in [284]) is used.

To sum up the various methods and techniques described above, the RL can be considered as comfortable tool of vehicle modelling. However, there are plenty of challenges occurring when modelling a multiagent system, related to joint actions. In the methodology of the current work, we should deal with these challenges.

2.3 Used methods and benchmarks

In this section, we discuss the chosen methodology for the use in the work as well as for benchmarking with developed in this work methods.

2.3.1 Empty vehicle redistribution

One important issue in taxi system management is how to minimise clients waiting time, and thus, to maximise the clients' utility. One of the problems related to this issue is the Empty Vehicle Redistribution (EVR) problem.

Based on the literature research, the EVR methods to use in the thesis can be divided in the following categories.

- **Reactive (Sending) algorithms**
  Basic allocation (BA): If there are waiting clients and empty vehicles at the same station, empty vehicles will be assigned to the clients with the longest waiting time (no redistribution of empty vehicles to other stations). This strategy is a baseline strategy and is used to evaluate the redistribution methods.
  Simple Nearest Neighbours (SNN): Reallocation of the nearest empty vehicles in order to serve the clients with the longest waiting time. This type of algorithms is used by Andréasson [14], Fatnassi et al. [90], Fagnant and Kockelman [88], and Lees-Miller [169].

Heuristic Nearest Neighbours (HNN): Reallocation of the nearest empty vehicles to serve the clients with the longest waiting time, taking into account the time it takes for vehicles to get to the clients. This method attempts to improve on SNN by taking into account the time it takes for a vehicle to move towards a waiting client. The difference between the SNN and HNN methods can be significant. Consider a network in a form of a long edge with one empty vehicle at one end of this edge and two clients waiting, one at each end of the edge. If the SNN algorithm is applied, the vehicle will move towards the currently longest waiting client, regardless of how long it takes to get to this client. On the other hand, if the HNN is applied, the vehicle will move to the other end of the edge because at the moment of its arrival the waiting time of the farthest client will be the largest. The discussion of this method is considered in Appendix A.2. This method is mentioned by Kek et al. [153], and Bell and Wong [31].

- **Proactive (Redistribution) algorithms**
  Surplus/Deficit vehicle Redistribution (SDR): The EVR from the station with the maximal vehicle surplus to the station with the maximal vehicle deficit. This method was used by Andréasson [14]. The vehicle surplus/deficit is defined for each station as the number of vehicles available (currently, or within a defined time horizon) minus the number of clients that are waiting (currently, or within the defined time horizon).

2.3.2 Dynamic ride-sharing

The objective of this subsection is to describe chosen methods of dynamic ride-sharing for autonomous taxi management systems. The popular single vehicle ride-sharing methods can be divided to two main families, described below.

In this subsection, we will use the following notations:
V is the number of vertices in the graph,
L is the characteristic path length,
D is the diameter of the city graph,
X is the total number of clients.

Searching on the way

This family of methods, referred to as "SOTW", aims to search for the additional client on or near the current vehicle trajectory.

If considering only the clients that are exactly on the current vehicle trajectory, the computational complexity of searching one new client is $\max(O(D), O(D \cdot \frac{X}{V}))$ and the memory consumption is $O(1)$. The main disadvantage of this method is that the method may miss the clients who can bring a small delay, but at the same time they can go absolutely to the same place as the original client. If the goal is to maximize the number of clients to the purposes of the ride-sharing, we should take into consideration not only trajectory-based clients, but also the clients placed somewhere near trajectory.

If considering the clients in a certain radius (or delay rate) of the current trajectory, 2 main methods can be highlighted:

Massive precalculation of the possible trip extensions for every 2 stations in the system takes time complexity $O(V^3)$ using Floyd-Warshall algorithm. It requires $O(V^2)$ of memory as for each of V stations we should keep list of size V of best neighbours to shortest path to other station. The preprocessing allows calculating the best time from one station to another with a complexity $O(1)$. Also, this method requires recalculation at the slightest change in the congestion. To find the best way to add just one client who wants to move from C to D to shared current trip from A to B, the cost of two trajectories $A \to C \to B \to D$ and $A \to C \to D \to B$ should be calculated. After the provided precalculation, the search of one client to add to the current trip takes $O(6 \cdot X) = O(X)$ time. Unfortunately, adding each next client to the general setting makes combinatorial variation of the

problem of travelling salesman with complexity of $O(X^N)$, where N is the number of clients to share, that is non-polynomial. The NP-hardness of the problem arises due to the fact that it is necessary to consider possible combinations of clients, and not just to add the "best" (according to some criteria) clients to the vehicle trajectory.

Let us provide an easy example of a gap between such a greedy and optimal solutions, as shown in Figure 2-5. Let the vehicle to be in point O and let us have a planned trip from O to D. Suppose that there are clients at stations 1, 2, and 3 willing to reach station D. When using a greedy method, the best suited client will add the less detour to the vehicle trajectory. After choosing such a client from the station 1, no more clients can be added to the trajectory.

If acceptable detour when ride-sharing is 10%, then the optimal ride-sharing strategy can be described as follows. Clients from stations 2 and 3 will be taken together on the same trip $O \to D$. This will bring reasonable delay within stated constraints.

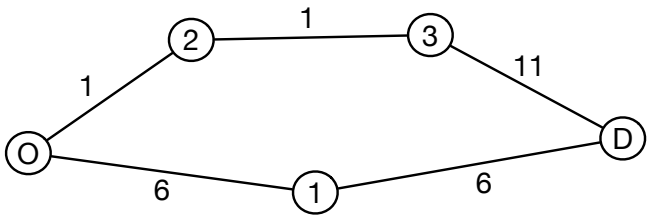


Figure 2-5 – Ride-sharing. Case 1.

The exhaustive search without precalculation. In this case to add one client to the current trip the exhaustive search takes $O(X \cdot V^2 \log E)$, using Dijkstra algorithm. Maximising the number of clients to share leads to more complex variant of problem from the previous paragraph. So the problem also belongs to the class of NP-hard problems.

Searching in the area of the origin and/or destination

This family of methods, referred to as "SOTA", aims to search for the additional clients that have origin and/or destination in the area of the current origin and/or destination.

Two main methods of such a search can to be highlighted.

Massive precalculation for each station of the system for all the nearest stations (and possible recalculations in the cases of the congestion), that takes $O(V^3)$ time and consumes $O(V^2)$ of memory (each station keep list of all other station with ascending order to time to reach). Let the number of stations in R-reachability of trajectory be Y and $Y = c \cdot L$.

As source and destination of possible ride-share trajectory must be in R-reachability set of stations, the search of one client to share takes $O(\frac{Y}{V} \cdot X)$ time complexity.

The search of N clients to share takes $O((\frac{Y}{V} \cdot X)^N)$ time complexity. The heuristics provide mostly greedy-based strategies, and, as in the previous subsubsection, can be far away from the optimum. When accepting clients in 2 minutes ride radius from the origin and/or destination, if there is a vehicle at station O and planned trip is from O to D, then the following example can be provided.

Let at Station 2 be 2 clients willing to reach Station 3, and at Station 1 to be 1 client willing to reach Station 4, as shown in Figure 2-6. Greedy method will assign to the current vehicle trajectory client from Station 1 as the closest both in terms of origin and destination. However the assignment of clients from Station 2 will not only make less detour, but will also help to serve more clients.

The exhaustive search without precalculation. To add one client to the current trip, the exhaustive search takes $O(V^2 \log E)$ time complexity, when maximising the number of clients to share the problem is NP-hard.

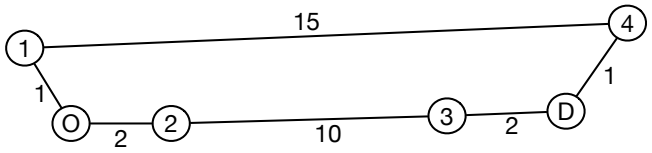


Figure 2-6 – Ride-sharing. Case 2.

This family of methods can also lose plenty of possible sharing scenarios. For example, let the current scheduled trajectory be from O to D, as shown in Figure 2-7. The client at Station 1 is willing to reach Station 2, and all its trajectory is exactly on the way of already scheduled one. However, this client will not be taken, because both its origin and destination are not in the area of 2 minutes ride from O and D.

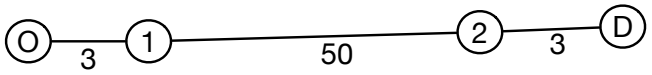


Figure 2-7 – Ride-sharing. Case 3.

2.3.3 Taxi management in complete information case

The problem of autonomous taxi system management can be considered as kind of congestion game because of limited number and given characteristics of vehicles. Congestion games [229] deal with strategic sharing of resource and appear in various domains, including network design and routing [161, 18, 251, 23]. The equilibrium search in specific games have received much attention [86, 193]. The gap between Nash equilibrium and optimum, called the price of anarchy, is one of particular research questions [230, 64]. The cost in this work is defined to be equal to the values of objective functions, notably to the number of clients served within their utility constraints 3.5.1 and, in case of equality, to the total duration of empty trips 3.5.2 from Subsection 3.5.1.

The search of the price of anarchy in different routing problems aims to find this value under different conditions. In ride-sharing systems, the objective is mainly the average load of vehicles, thus the value of price of anarchy ratio can be bounded by 2 using particular set of methods [98]. The stable matchings are basis for a lot of vehicle management systems, because researchers focus on non-cooperative games [24, 99].

The case of complete information is the study of system performance a-posteriori. That is, the goal is to determine how vehicles should act if all arrivals and preferences of clients are known in advance. Thus, the global optimum of the system can be found.

Even in the case of not ride-shared systems, this problem belongs to the class of NP-hard problems. The search of system (social) optimum in taxi systems is provided by heuristical methods. Pareto-efficient services provide mostly only local optimums, like the methods studied in centralised learning model, and are mostly based on matchings. The matchings between different agents and the bilateral search is one of classical questions in economics [201, 212, 227]. The matching function was first introduced in 1983 [237] to analyze taxi services in a regulated market, where the main modes of operation were radio control and taxi parking at the airport. The different approaches of using and evaluation of social optimum are considered in the literature. The comparison of social optimum in taxi services under conditions of monopoly and perfect competition under the hypothesis of constant returns to scale is provided in [47]. The changes in taxi regulations can be evaluated by searching and meeting frictions [164, 165]. The Markov chain approach is also used for a stochastic bilateral search in taxi systems [289]. Yang et al. uses a meeting function to prove the existence of stationary competitive equilibrium achieved at fixed fare prices in taxi market. Sirisoma et al. [242] provide

empirical evidence basing on 400 taxi drivers for the study urban taxi services under the customer-taxi search model [290, 291, 299, 300].

The congested taxi market eliminates the marginal effects when scaling [302], thereby the entry of additional vehicles and the taxi-fleet in general should be highly controlled. In [301] the Pareto-improving service is provided, showing the returns to scale in taxi systems.

The subproblem of the electrical taxi management problem without ride-sharing and chargings can be simplified to the ACVRP (CVRP with asymmetric cost matrix), that, in not-dynamic version, has the following methods of the solution, both based on the branch and bound approach. The ACVRP was solved in the literature using lower bound [166] based on the Assignment Problem (AP) relaxation or the AP lower bound with a lower bound based on disjunction and one based on a min-cost flow relaxation [95]. However, even with all the updates in this class of methods for this problem, the problem stays NP-hard and the maximal number of consumers stays in the order of one hundred. Thus, in the opinion of researchers [264], for the set of specific networks, these approaches can help to solve only problems limited to few tenths of customers.

One of possible exact methods for the simplified problem solution (with non-stochastic clients, without vehicle size constraints, etc.) is based on the following considerations [240]. Authors define the EV fleet routing problem on a complete directed graph and formulate the EVRP-RS as a mixed integer nonlinear programming (MINLP) problem. The formulation not only has a compact two-index form but also takes EVs' multiple visits to the same charging station into account. After, authors aim to transform the MINLP problem into an equivalent MILP problem where global optimum can be found in many instances. However, the proposed method does not take into account possible ride-sharing detours, and the charging methods does not depend on the demand in the system. Ride-sharing with possible detours was, for example, considered in the problem of matching driver-rider [254].

2.4 Conclusion

The user preferences research over waiting and travel times uncovers that their utility function should not necessarily be linear. The electric vehicle charging and discharging research shows that in the modern literature for both microscopic and macroscopic modelling optimisation problems, the kilometre discharge is mainly used as discharge function, and the charging is considered to be linear by time. In this thesis we will use the same type of charging and discharging functions with the parameters obtained from the literature.

The empty vehicle redistribution research shows that there are two main methods: reactive, where the vehicle is being sent to the passenger when passenger arrives, and proactive (rebalancing), where the vehicle is being sent to another region of the city to meet the demand, may be predicted one. Our models will try to accommodate both types of empty vehicle redistribution.

The ride-sharing methods research reveals that even the methods that are already existed (even the simplest ones, when only passengers with the same origin and destination will be picked up with the same vehicle) can significantly improve economical and ecological system performance. In this work we will propose new methods of ride-sharing based on zone clustering and machine learning approaches.

A review of the machine learning methods show that for the problem setting and research questions of the current work the best suited methods are the follows:

1. Clustering methods for city partitioning, demand modelling and simulation and speed prediction.
2. Reinforcement learning methods for taxi simulation as a general problem.
3. Differential evolution genetic learning method to obtain optimal weights for RL coefficients.

Chapter 3

Modelling the electrical aTaxi system

3.1 Introduction

An important element in the concept of "smart city" is an efficient operating transport system of a high level, which includes autonomous vehicles. The use of autonomous vehicles allows us to increase the mobility of moving people and to reduce the number of vehicles and parking lots. In the long term, it can reduce travel time and increase safety by reducing human errors related to quick evaluation of large number of objects such as vehicles, pedestrians, or road transport system.

In this chapter, the model of electric autonomous taxi system is described, as well as the optimisation problems occurring in such a system. This chapter is structured as follows.

Section 3.2 describes the framework of the work, including the system description. Next sections make a transition from the system itself to the model of the system. The components of the model can be divided into two main groups. The components of the first group, called static ones, are described in Section 3.3, and are defined from the setting of the system and represent elements such as the network structure, the number and location of the charging stations with their parameters, the number of vehicles in the system and the demand patterns. These elements do not depend on vehicle movements. The components of the second group, said dynamic and described in Section 3.4, are specific parameters for this model and represent the information that change depending on the vehicle movements.

The optimisation problem formulation for the electric aTaxi system is given in Section 3.5. According to its complex structure, a set of subproblems can be highlighted. The main subproblems aborded in this thesis are the empty vehicle redistribution problem, the dynamic ride-sharing implementation and charging strategies. Formulations of these subproblems are provided in Subsections 3.5.2, 3.5.3, and 3.5.4.

3.2 System description and hypothesis

In this section we describe the framework elements of the considered system. The main dependencies in this framework are shown in Figure 3-1.

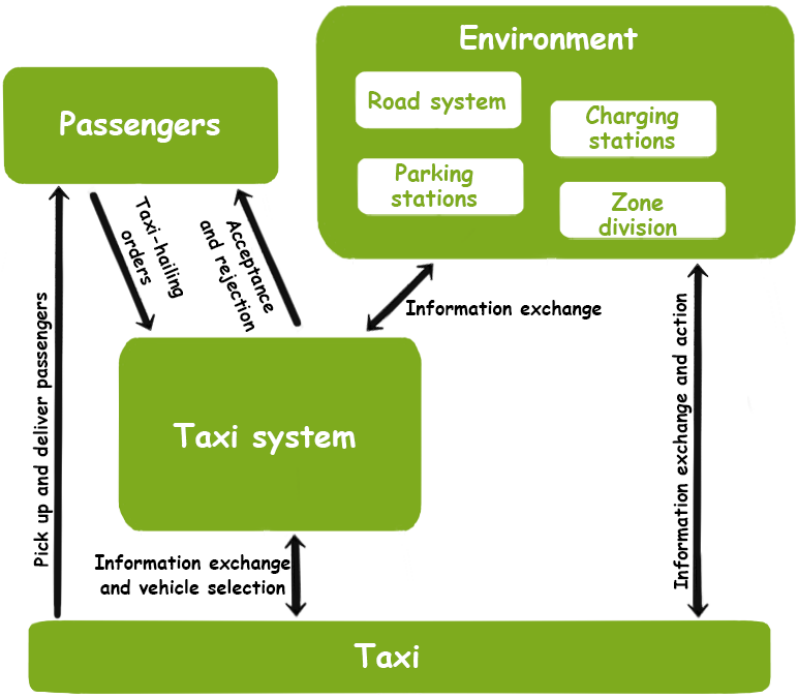


Figure 3-1 – aTaxi system structure.

3.2.1 Environment

Electric aTaxi system is a part of urban infrastructure with given road system, charging and parking stations, train stations and bus stops, with various establishments of shopping, education, entertainment, etc. In the terminology of this work, taxi stations are defined as places where the vehicle can wait for the next command from the system dispatcher during a given time interval. Charging or parking stations can also be taxi stations. Boarding and disembarking passengers is possible only at taxi stations. Hereinafter, we refer to the taxi stations as to "stations".

Vehicles can only move along a given network of roads. The standard taxi system for ordinary cars in most cases assumes that vehicles arrive very close to a specific object such as a residential building or establishment. In the case of autonomous cars use, expanding the capabilities of the system in this direction is possible at the next stage, when the operation of the system will be sufficiently studied for the case of boarding and disembarkation only at given stations, however, located quite closely.

For a more accurate and faster modelling, we propose the simulation of a city road network divided in zones.

In Figure 3-2 is shown one of possible methods of the city road network division on the example of the city of Paris. This method is based on the administrative districts. Nevertheless, other types of the city division by zones, including overlapped ones, are described in this chapter.

3.2.2 Passengers and demands

The passenger's goal is to move from one station to another in the shortest possible time, which may include, in some desired ratio, the waiting time and travel time. Thus, the efficient service for passengers can be determined by the number of passengers served within a given time interval, and by the vehicle waiting time. The standard passenger request in taxi systems includes the departure station, the arrival station, and the desired departure time (if the vehicle is not needed right now). A more detailed description of passenger requests in the framework of the proposed model is presented below.

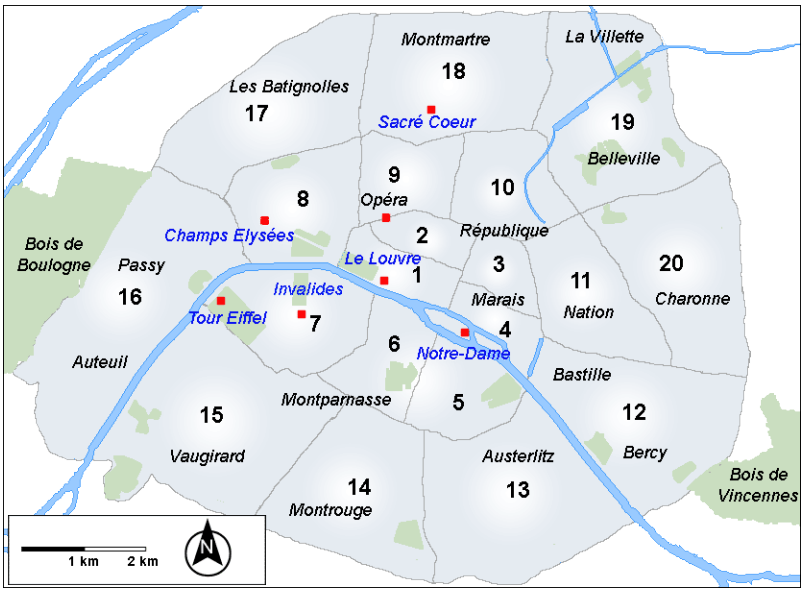


Figure 3-2 – Arrondissements of Paris.

3.2.3 Taxi system

The purpose of the taxi system is to effectively serve passengers arriving in this system, that is, passengers who called a taxi for now or for a specific time (later). The autonomous taxi system consists of taxi stations, autonomous vehicles and a control program-dispatcher.

We consider a system of autonomous electric taxis with the predefined parameters, which move in the environment between the stations. This system offers a taxi with the option of ride-sharing. That is, the passenger, if he/she wishes, can agree to share his/her trip with other passengers. This approach potentially allows serving more passengers with a fixed fleet size.

A specific vehicle can serve a request only under the following necessary conditions: the vehicle must have enough charge to reach the passenger in its current position, deliver him/her to the planned destination and then reach the nearest charging station. *What is an instant snapshot of the system at the time a request is received from the client?* Some of the vehicles are at the stations, some are travelling to passengers, some are delivering them to the desired stations, some are being charged or parked, and some are travelling to or from charging and parking stations. For each road section, the expected travel time for a given traffic situation and/or time of day is known. Thus, for each vehicle in the system, the expected travel time to the specified station of the calling passenger can be calculated.

3.2.4 Objectives

The development of an autonomous taxi system that works efficiently in a particular city can be represented as two interrelated problems.

The first problem is to determine the number of stations, to distribute them on a city map, and to determine the number of service vehicles. In the provided approximation, the solution of this problem is based on the analysis of passenger traffic related to public transport and private vehicles before the arrival of autonomous taxis in the city.

The second problem is to provide efficient passenger service with given taxi fleet and stations. If the solution of this problem is far from ideal, even with optimal driving, then perhaps the reason is the lack of stations or vehicles. If a large percentage of vehicles is idle, then it is possible that there is a surplus of them, which brings us back to the first problem. The main purpose of this work is to solve the second problem.

3.2.5 Hypothesis

In this subsection, we describe our different hypotheses occurring when modelling the electric autonomous taxi system.

Our hypotheses are based on existing theories (see Chapter 2) as well as represent new not-studied yet theories.

- The implementation of the ride-sharing can improve the system performance. It is especially effective to use ride-sharing in case of pre-ordering the taxis.
- The empty vehicle redistribution can significantly improve system performance.
- The methods of empty vehicle redistribution and ride-sharing which are based on road graph clustering can accelerate optimisation without loss of system performance.
- Machine learning methods can allow to study the preferences of passengers.
- In the event of a change in system parameters, machine learning methods can show greater adaptability in the long run.
- In the case of a predefined traffic conditions, methods of reinforcement learning do not show a significant superiority compared to classical optimisation methods, including methods based on differential evolution.

3.3 Static components of the model

In this section we introduce the static components of the model of electrical autonomous taxi system management and the necessary notation used. These components include definition of the environment, the demand patterns, and the vehicle fleet with its characteristics.

The time in the model is noted t and considered to be discrete with the small step of discretisation equal to one second.

3.3.1 Environment

In this subsection, we describe the main components of the environment of the system, including road network, zones, and parking and charging stations.

Network representation

The real network is in general represented by set of roads and road intersections. There are two main different representation methods for urban street networks, called the primal approach and the dual approach.

The primal representation is a natural and intuitional approach, which takes just the road segments as edges, and intersections or the ends of roads as vertices for an urban street network. Because of the coherence between the dimension of geographic and graph entities, this kind of representation is hereby termed "direct", or primal.

In contrast, the dual ("indirect") representation could introduce more subjectivity. In this representation streets are turned into vertices and intersections are turned into edges, and if 2 streets intersect, the edge between them will be constructed. In this sense, defining the streets (i.e., the meaning of the vertices) can cause problems such as what exactly one street is and if this definition depends on the official street name or some geographical properties.

Let us have a city street network primal representation [214] as a directed weighted graph $G = G(V, A)$, where

each vertex $v \in V$ is a road intersection and/or a taxi station in a street. We consider the system with station-to-station taxis, thus every origin and destination points of passengers are vertices of the graph.

each arc $a \in A$ is defined as $a = (init_a, ter_a, distance_a, speed_a(t))$, where $init_a \in V$ and $ter_a \in V$ are the initial and terminal points of arc a, respectively. The speeds on arcs are time-dependent functions, with constant expectancy for every day and day period. The arc from vertex v_1 to vertex v_2, where $v_1, v_2 \in V$, can be referred to by $a_{v_1 v_2}$. Because of client walking distance implemented in this system, each arc a uses the distance $distance_a$ and speed $speed_a$ characteristics instead of just a time to pass. When entering the arc, the speed of crossing it is considered to be constant, so the time to pass the arc can be defined and calculated as $time_a(t) = \frac{distance_a}{speed_a(t)}$.

Based on the network representation, for several possible subproblems occurring in electric aTaxi system, the alternative simplified representation of the city road network is possible. In the OD time matrix representation, given a set of vertices V, for each ordered pair of vertices v_1, v_2 at every start time t is given the minimal travel time $T[v_1, v_2](t)$ from v_1 to v_2. We assume the travel times to be constant for each optimisation time interval.

Zone

The city in the system is divided to a set of zones. Subsequently, we consider that the road network graph in the model is divided to a set of zones Z.

Each zone $z \in Z$ is a strongly connected subgraph of G induced by a subset of vertices. This zone can be obtained as the subset $V_z \subset V$ of the graph which arcs A_z between vertices belonging to V_z belong to the initial graph G.

The set of all zones covers all the vertices of the city road graph: every vertex of this graph belongs to at least one zone. The zone clustering methods and detailed description of such a structure are described in Section 4.2.

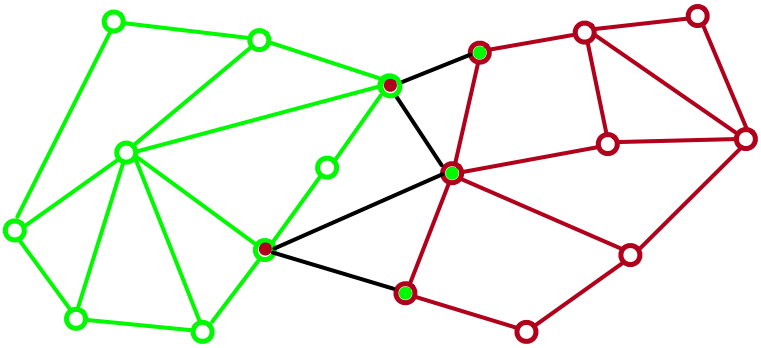


Figure 3-3 – Zone clustering.

In Figure 3-3 the green and red parts represent two different zones, and the black edges belong to both zones simultaneously and represent zone overlapping.

Stations

In the system there are 2 different types of stations: charging stations and parking ones.

For each station $\gamma \in \Gamma$, we have the following characteristics:

The position is the vertex $v_\gamma \in V$ where the station is located. Each station is associated with exactly one vertex of the network graph. We can consider this vertex as an entrance to the station.

The capacity S_γ^{max} is the maximal number of vehicles the station can contain.

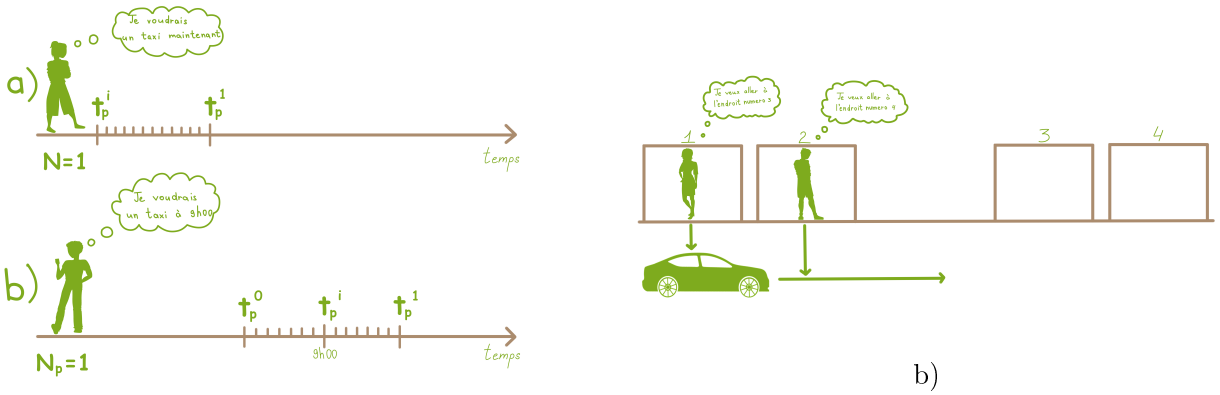
The type $Type_\gamma$ is the discrete value in the set $\{0; 1\}$, where 1 represents a station with recharge plots and 0 a parking station.

For every zone $z \in Z$, the set of stations (parking and charging stations) is defined as $\Gamma_z \subset \Gamma$.

3.3.2 Demand

This subsection is intended to describe the demand in the system, that is the clients with all their characteristics.

In the model, the set of passenger groups is denoted by P. Each passenger group $p \in P$ (from this point we refer to it as "client") that arrives to the system can be characterised by the following parameters:



a) b)

Figure 3-4 – a) Time windows b) Ride-sharing.

Number of passengers N_p: we consider that the size of the passenger group is smaller or equal to the capacity of the vehicle.
Origin $v_p^o \in V$ is the pick-up point of client p.
Destination $v_p^d \in V$ is the drop-off point of client p.
Time window is represented by ordered pair of times (t_p^0, t_p^1), where t_p^0 and t_p^1 are the minimal and the maximal times of picking up, respectively. The time that embarkment and disembarkment takes is not included to the time window. To have the possibility to serve a client, the constraint $t_p^0 < t_p^1$ has to be satisfied. The time window size of the client is defined as $TW_p = t_p^1 - t_p^0$.
Ideal pick-up time t_p^i: this number refers to the ideal time for picking up client p. For clients willing to take a taxi immediately when calling, this number is equal to t_p^0. In the model, the condition $t_p^i \in [t_p^0, t_p^1]$ has to be satisfied.
Readiness to share $rtosh_p \in \{0; 1\}$, is equal to 0 if client p is not ready to share the ride and 1 if it is ready.
Readiness to walk w_p is the maximal distance client p agrees to walk to take a vehicle. We consider the client generation at stations, but the distances between stations are also given. Thus the client can reach one of nearest stations on foot.
Client call time t_p^{app} is the time of client's taxi call. From this moment the system is aware of client p existence.
Utility constraint u_p^{min} is the minimal utility value for the trip accepted by client p. The utility of the client p is described in Subsection 3.4.6.

For every client in the system there is a globally shared constant $Ratio_{max}$, the total travel time to minimal possible driving time ratio.

3.3.3 Vehicle

This subsection describes the vehicle characteristics. Each vehicle $\xi \in \Xi$ can be characterised by the following constant parameters:

The capacity S_ξ is the maximal number of passengers that can be in the vehicle.
The charging information is the set of the following parameters:

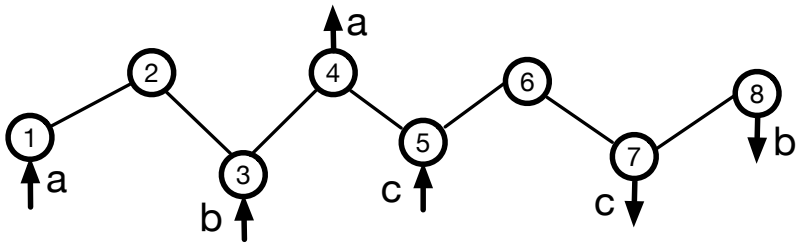


Figure 3-5 – A vehicle trajectory example.

— *Maximal travel distance on one charge of the battery* D_{ξ}^{max}. We consider that all the vehicles in the system discharge linearly with a constant rate, which does not depend on total weight of passengers inside, but on the distance travelled. During the parking time the vehicle does not discharge.
— *Charging time from empty to full battery* Ch_{ξ}. We consider that vehicles charge linearly with a constant rate on time and that all the charging stations in the system are identical in terms of charging characteristics.

3.4 Dynamic components of the model

The system changes with time: the vehicles move from one position to another and discharge. The clients enter and leave the system, being satisfied or not. In order to provide necessary characteristics for the problem solution and its evaluation, in this section we provide the dynamic components of the system. We remind, that the static components are fixed at problem setting initialisation, whereas the dynamic ones will be changed with time.

3.4.1 Environment

The environment of the model as the vehicle moves, and the dynamic components of the environment take these vehicle movements into account.

Each **zone** $z \in Z$ in time t has the following dynamic parameters:
Set of vehicles $\Xi_z(t)$ in this zone;
Demand in this zone is the set of clients $P_z(t)$ who have already booked a taxi.

Each **station** $\gamma \in \Gamma$ has the following dynamic parameters:
Current occupancy $S_{\gamma}(t) \leqslant S_{\gamma}^{max}$ is the number of vehicles located at station γ at time t.
Other vehicles arrivals: every station can be also used as station for other vehicles in the system. For every station γ there is an empty slot probability $P_{\gamma}^{available}(t)$ which depends on t, that includes the day type and the day period.

3.4.2 Trips in the system

Every vehicle in the system has a travelling trajectory. Every vertex passed by the vehicle is a "passing vertex" or a "vertex with event", where the event is picking up/dropping off a person, start or end of charging cycle or start or end of vehicle parking.

Figure 3-5 illustrates a travelling trajectory of a vehicle. In this example, client a boards in vertex 1 and disboards in vertex 4, client b boards in vertex 3 and disboards in vertex 8, and client c boards in vertex 5 and disboards in vertex 7. Therefore, the event vertices are 1, 3, 4, 5, 7 and 8.

In the model, we define a "trip" as a part of the trajectory between 2 vertices with events. Each vehicle is associated with its trip sequence, that is referred to as "vehicle trajectory", and each client is associated with its trip sequence, that is referred to as "client trajectory", or "ride".

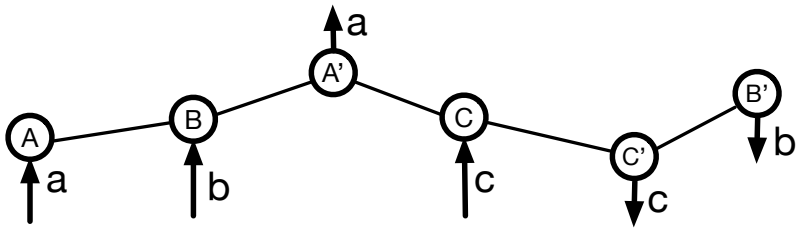


Figure 3-6 – A trip sequence example.

The trips in the model can be considered both from client and vehicle perspectives. The trip $r \in R$ can be defined univocally from the information about its origin $v^o(r)$ and its destination $v^d(r)$, and the vehicle ξ_r carrying out the trip of one of the clients p travelling.

Each trip $r = r(v^o, v^d)$ contains:

Origin vertex $v^o(r)$,

Destination vertex $v^d(r)$,

Set of clients inside the vehicle during the trip $P(r)$. Any client p of the trip can be referred to using the upper index r^p.

Vehicle carrying out the trip $\xi(r)$. The vehicle ξ of trip r can be referred to using a lower index r_ξ.

Time to finish the trip $tend(r)$ characterises univocally the current position of the vehicle.

3.4.3 Vehicle

The dynamic parameters for each vehicle $\xi \in \Xi$ are the following:

Position of the vehicle ξ at a given moment t is defined by the current arc of the vehicle $a_\xi(t)$ as well as its position $l^a_\xi(t)$ on this arc.

Current set of assigned trips (assigned trajectory) $R_\xi(t)$ is the sequence of trips of the vehicle. The next trip starts immediately after the finish of the current one.

Current charging $D_\xi(t)$ is defined as a distance the vehicle can travel from its current position, before needing to recharge.

3.4.4 Client

The dynamic parameters of client $p \in P$ are the following:

Departure time t^{dep}_p is a parameter known after a simulation and refers to the real departure time of the client. In case where the client is lost for the system (for example, because its call was not accepted) this time is undefined.

Assigned trip sequence R^p is an assigned ride to client p (see Subsection 3.4.5).

Walking distance w^{real}_p is the real distance client p walked to take a vehicle.

3.4.5 Trips from client and vehicle perspectives

For a better representation of trips from both client and vehicle perspectives, let us consider the example in Figure 3-6, constructed based on Figure 3-5.

Client a starts at point A and finishes at point A', client b starts at point B and finishes at point B', client c starts at point C and finishes at point C'. There is only one vehicle that serves all the clients. Thus, for the trip sequence in Figure 3-6, the client rides are the following:

$R^a = \{r^a(A, B); r^a(B, A')\}$,
$R^b = \{r^b(B, A'); r^b(A', C); r^b(C, C'); r^b(C', B')\}$,
$R^c = \{r^c(C, C')\}$.

For the trip sequence in Figure 3-6, the vehicle trajectory R_ξ is defined as

$R_\xi = \{r_\xi(A,B); r_\xi(B,A'); r_\xi(A',C); r_\xi(C,C'); r_\xi(C',B')\}$,
where the corresponding sets of clients in each trip are the following:
$P(r_\xi(A,B)) = \{a\}$,
$P(r_\xi(B,A')) = \{a;b\}$,
$P(r_\xi(A',C)) = \{b\}$,
$P(r_\xi(C,C')) = \{b;c\}$,
$P(r_\xi(C',B')) = \{b\}$.

3.4.6 Client utilities

The client utilities in the model are used to know, if the client will accept the proposed ride, as well as the satisfaction of the client after accepting and performing the proposed ride.

The expected quality of the system service from the client perspective can be described as function of:

- the waiting time (time from the desired departure time until the real departure time);
- the ride-sharing (the presence of the ride-sharing in the trip and delay due to ride-sharing);
- the real walking distance w_p^{real}.

We consider 2 different models:

— A classical approach, where the client is considered to be satisfied if it is taken within its constraints and considered to be not satisfied if any of the constraints is not respected.

— A stochastic approach, where client is considered with non-linear preference functions and randomisation within the utility constraints. The taxi system itself is not aware of the utility function preferences, but receives responses from clients if they are accepting the trip or not, or if they are satisfied by the trip performed or not. The non-linear and stochastic nature of the preference functions helps us to model better the real human behaviour. However, the utility of the client is used only as discrete decision of its satisfaction.

Proposed ride acceptance condition

The system or the vehicle itself proposes the ride to the client. The proposed ride must satisfy the parameters such as readiness to share, readiness to walk and time-window of the departure. The system proposes the arrival time of the taxi.

Utility of client $p \in P$ is a number in $[0;1]$. The client accepts the proposed ride if the characteristic value of this ride is bigger than utility constraint value u_p^{min}.

The characteristic value of proposed client ride can be calculated as function of:

- the proposed walking distance w_p^{prop},
- the proposed taxi arrival time t_p^{prop},
- the ride-sharing existence of the proposed ride rsh_p^{prop}.

The proposed ride characteristic value of these terms can be calculated as described below.

Hereinafter 0 is characteristic value for the client whose ride is not within its constraints, whereas 1 is characteristic value for the client whose proposed ride is exactly in its desire.

Proposed waiting time characteristic value The proposed waiting time characteristic value of client p is noted $u_{waiting}(t_p^{prop}, t_p^i, t_p^0, t_p^1)$, where t_p^0, t_p^1 are the client's time window constraints and t_p^i is the ideal arrival time of the taxi. This function can differ depending on clients time window. The client that is not served within the time window is lost for the system and has utility equal to 0.

In *deterministic case*, this characteristic value function is defined as:

$$u_{waiting}\left(t_p^{prop}, t_p^i, t_p^0, t_p^1\right) = \begin{cases} 1, & \text{if } t_p^{prop} \in \left[t_p^0; t_p^1\right]; \\ 0, & \text{if } t_p^{prop} \notin \left[t_p^0; t_p^1\right]. \end{cases}$$

In *nonlinear stochastic case*, function $u_{waiting}(t_p^{prop}, t_p^i, t_p^0, t_p^1)$ is nonlinear and can be defined as:

$$u_{waiting}\left(t_p^{prop}, t_p^i, t_p^0, t_p^1\right) = \begin{cases} 1, & \text{if } t_p^{prop} < t_p^i; \\ 1 - \left(\dfrac{t_p^{prop} - t_p^i}{t_p^1 - t_p^i}\right)^3, & \text{if } t_p^{prop} \in \left[t_p^i; t_p^1\right]; \\ 0, & \text{if } t_p^{prop} > t_p^1. \end{cases}$$

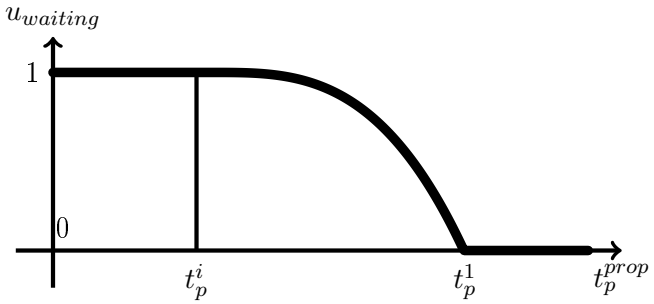


Figure 3-7 – Waiting time characteristic value function shape.

There are some constraints that client cannot take a taxi before the time-window. If the vehicle arrives earlier than the beginning of the client's time window t_p^0, the proposed boarding time from the point of view of the client will be equal t_p^0. This explains the shape of the proposed curve: the client considers the proposed taxi arrival time as the minimal time he can board, without strong requirements to board before its ideal time. Further, the client make not a big difference between proposed boarding times near its desired boarding time, and after every waiting minute makes him not linearly unhappy, reaching 0 by the end of its time window.

Proposed walking characteristic value The characteristic value $u_{walking}(w_p, w_p^{prop})$ of proposed walking depends on the proposed walking distance w_p^{prop} and the maximal accepted walking distance w_p. In the model, it can be modelled with the same type of functions than proposed walking characteristic value function.

In *deterministic case*, this function is defined as follows:

$$u_{walking}(w_p, w_p^{prop}) = \begin{cases} 0, & \text{if } w_p^{prop} > w_p; \\ 1, & \text{if } w_p^{prop} \leqslant w_p. \end{cases}$$

In *nonlinear stochastic case*, the function is defined as:

$$u_{walking}(w_p, w_p^{prop}) = \begin{cases} 0, & \text{if } w_p^{prop} > w_p; \\ 1 - \left(\frac{w_p^{prop}}{w_p}\right)^3, & \text{if } w_p^{prop} \leqslant w_p. \end{cases}$$

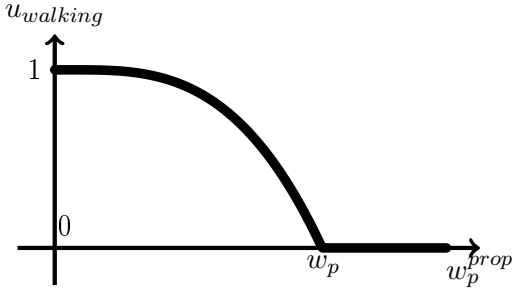


Figure 3-8 – Walking distance characteristic value function shape.

The shape of this nonlinear curve is explained similarly to the situation above: the client considers its walking not linearly, and every additional meter makes it more unhappy than previous one.

Proposed ride-sharing characteristic value function The characteristic value function of the ride-sharing $u_{rsh}(rtosh_p)$ is the stepwise function, and is equal to 0 in case of proposed ride with ride-sharing to the client who does not want to share the ride and 1 otherwise.

Thus, the client that does not accept shared ride will never accept it even if it will not have to wait or to walk in this case.

Proposed ride characteristic value function The total characteristic value of the proposed ride R is a product of the proposed utilities:

$$u_{proposedride}(p, R) = u_{waiting}(t_p^{prop}, t_p^i, t_p^0, t_p^1) \cdot u_{walking}(w_p, w_p^{prop}) \cdot u_{rsh}(rtosh_p).$$

The ride will be accepted by the client if $u_{proposedride}(p, R) \geqslant u_p^{min}$. The system does not know the client's utilities and learns them with the time.

The performed ride satisfactory condition

When performing a ride R, the characteristic value of client p ride R^p can be described as combination of the following parameters:

$u^R_{waiting}(t_p^{real}, t_p^{prop}, t_p^0, t_p^1)$: characteristic value function of real waiting time depending on real taxi arrival time t_p^r, and on client time-window and predicted by taxi service pick-up time. Client that is not taken within the time-window is lost for the system and has 0 utility. If the proposed vehicle arrival time is earlier than t_p^0, t_p^{prop} becomes t_p^0.

$u_{delay}(R^p)$: delay characteristic value for client p sequence of trips (ride) R^p.

Performed ride waiting characteristic value The characteristic value of real waiting time can be calculated the following way:

In *deterministic case*, the function is defined as follows:

$$u^{real}_{waiting}(t_p^{real}, t_p^{prop}, t_p^0, t_p^1) = \begin{cases} 1, & \text{if } t_p^{real} \in [t_p^0; t_p^1]\,; \\ 0, & \text{if } t_p^{real} \notin [t_p^0; t_p^1]\,. \end{cases}$$

In *nonlinear stochastic case*, the function is defined as:

$$u^{real}_{waiting}(t_p^{real}, t_p^{prop}, t_p^0, t_p^1) = \begin{cases} 1 - k_{early}, & \text{if } (t_p^{real} < t_p^0) \bigcap (P(r) = \emptyset); \\ 0, & \text{if } (t_p^{real} < t_p^0) \bigcap (P(r) \neq \emptyset); \\ 0, & \text{if } t_p^{real} > t_p^1; \\ 1 - k_{late} \cdot \left(\frac{t_p^{real} - t_p^{prop}}{t_p^1 - t_p^{prop}}\right)^3, & \text{if } t_p^{real} \in [t_p^{prop}; t_p^1]; \\ 1 - k_{early} \cdot \left(\frac{t_p^{prop} - t_p^{real}}{t_p^{prop} - t_p^0}\right)^3, & \text{if } t_p^{real} \in [t_p^0; t_p^{prop}]. \end{cases}$$

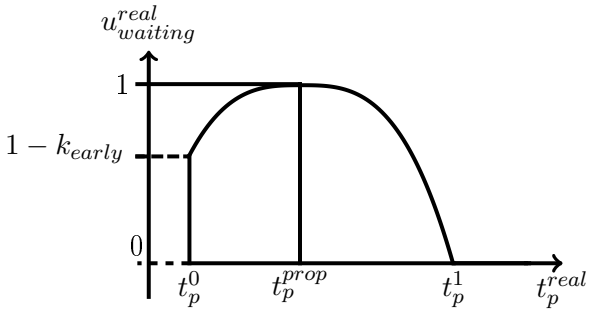


Figure 3-9 – Performed ride waiting time characteristic value shape.

Coefficients k_{late} and k_{early} show asymmetry for late and early departure cases; $k_{late} > k_{early}$ as early departure is generally more preferable for clients, but also there are some constraints that client can not take a taxi before its time-window.

To simplify the calculations, we let $k_{late} = 1$, and $k_{early} = \frac{1}{2}$.

The client will take the vehicle if it comes within the time window. In case, where the vehicle comes before moment t^0_p, the vehicle can wait the client (who will board on at time t^0_p. In the case of shared ride and some clients already inside the vehicle, this one can decline the call of the client and this corresponds to the 0 of characteristic value). The condition $P(r) \neq \emptyset$ refers to the presence of the clients inside the vehicle on arrival.

Performed ride delay characteristic value Let TTT_p be Total Trips Time of client p, thus, the total time from the boarding to the disembarking of the client p.

Let $MPDT_p$ be minimal possible driving time for client p, thus, the minimal possible driving time (based on current congestion level) from v^o_p to v^d_p.

The delay characteristic value for the client's sequence of trips can be calculated as the total delay in the way and is defined as follows:

$$u_{delay}\left(TTT_p, MPDT_p, Ratio_{max}\right) = f_{delay}\left(\frac{TTT_p}{MPDT_p}\right).$$

In *deterministic case* it corresponds to:

$$f_{delay}(x) = \begin{cases} 0, & \text{if } x > Ratio_{max}; \\ 1, & \text{if } x \leqslant Ratio_{max}. \end{cases}$$

In *nonlinear stochastic case* this function is defined as:

$$f_{delay}(x) = \begin{cases} 0, & \text{if } x > Ratio_{max}; \\ 1 - (\frac{x}{Ratio_{max}})^3, & \text{if } x \leqslant Ratio_{max}. \end{cases}$$

Characteristic value of the performed ride The value is defined as the following product of characteristic values of waiting time and delay on ride due to ride-sharing.

$$u^R_p = u^{real}_{waiting}(t^{real}_p, t^{prop}_p, t^0_p, t^1_p) \cdot u_{delay}(TTT_p, MPDT_p, Ratio_{max}).$$

If this characteristic value is less than the client's utility constraint, the client is not satisfied by the system. The system has critics from the client (for example, as a note in the application or bad review) and its utility is considered to become 0. When calculating the system performance we consider such clients to be equivalent to clients that are not served and are lost for the system.

3.5 Optimisation problem definitions

The inputs of the model are:

The graph $G(V, A)$ that models the city network with set of the stations and their characteristics.

The predefined demands P in the form of clients with all the attributes such as number of passengers, origin, destination, time window, readiness to share and ideal pick-up time. Note that client p is recognised by the system at the moment of the taxi call t_p^{app}.

The vehicle set Ξ with their charge levels and start positions.

The output of the model is the set of all the vehicle trajectories R_ξ, $\forall \xi \in \Xi$, which lead also to all the client rides information.

The expected quality of the system is the synthesis of expected quality from both client and service perspectives. The expected quality from the client perspective can be described as function of the utilities of the trip (thus, characteristics such as waiting time, walking distance and ride-sharing parameters). The expected quality from the service perspective can be described as function of the total number and the duration of the trips, and the number of clients in all empty and non-empty trips. In this section firstly we define the optimisation problem and after we formulate the related subproblems. The inputs of these subproblems are identical.

3.5.1 Dynamic aTaxi management problem setting

The expected quality of the system is considered as a combination of sets of objective functions.

The first objective is to maximise the number of clients served within their utility constraints with fixed taxi-fleet,

$$\max \sum_{p \in P} F_p, \tag{3.5.1}$$

where $F_p = 1$, if client p was served within its utility constraints and $F_p = 0$ if it quitted the system without being served.

The second objective is to optimally consume the energy, thus to minimise the duration of empty trips

$$\sum_{\xi \in \Xi} \left(\sum_{r \in R_\xi, P(r)=0} L_r \right) \to \min, \tag{3.5.2}$$

where L_r is the length of trip r.

The third objective is to minimize the fleet size to serve a certain percentage of clients in the system. In order to solve this problem, the following set of subproblems can be formulated:

1. Charging management problem.
2. Empty vehicle redistribution problem.
3. Dynamic ride-sharing problem.

Below, the formal definitions of these problems are represented.

3.5.2 Charging management problem

The objective of charging management in the system is to reach an equilibrium between chargings and movements in the system. The objective cannot be formulated as just maximisation of charging levels because in such a formulation the vehicles will not serve clients being always on charging stations. The objective is then to serve clients minimising the number of uncharged vehicles. If considering that uncharged vehicle will leave the system, this objective can be reformulated as maximisation of non-empty vehicle trips over the simulation time, thus, is given by Formula (3.5.2).

3.5.3 Empty vehicle redistribution problem

The objectives of the empty vehicle redistribution problem with time-windows can be formulated such as follows:

a) To maximize the number of clients served within their time windows, as defined by optimisation Formula 3.5.1 in Subsection 3.5.1.

b) To minimize the average and maximal waiting times of served clients

$$\min \frac{\sum_{p \in P, F_p=1}(t_p^{real} - t_p^0)}{|p \in P, F_p = 1|}, \tag{3.5.3}$$

$$\min \max_{p \in P, F_p=1}(t_p^{real} - t_p^0), \tag{3.5.4}$$

where t_p^{real} is real departure time of client p.

Different methods can show optimums for different objective functions. The optimum obtained in terms of maximal waiting time does not have to lead to minimal average waiting time. The following simplification of real-life taxi service is considered. The client does not need immediate report from the taxi service. The client is considered to be satisfied if it is taken within its constraints and considered to be not satisfied if any of the constraints is not respected. The client will quit the system at the end of its time window even if its taxi is already approaching.

3.5.4 Dynamic ride-sharing problem

In the problem of dynamic ride-sharing, the expected qualities can be simplified as follows. The expected quality from the client perspective can be described as function of the number of served clients (within their utility constraints). The expected quality from the system perspective can be described as function of the number of served clients and the number of clients in shared trips.

Thus, the first objective in the problem is the optimisation problem defined by Formula (3.5.1) in Subsection 3.5.1.

The general objective in this problem can be formulated as follows. To maximize the function represented as the total number of served clients divided by their delays caused by ride-sharing

$$\max \sum_{p \in P} \frac{F(p)}{D(R^p)}, \tag{3.5.5}$$

where $F(p) = 1$, if client p was served and $F(p) = 0$, if client p was not served until the end of the simulation. $D(R^p) = \frac{TTT_p}{MPDT_p}$, where TTT_p is the real time the client spends in the way, and $MPDT_p$ is the shortest possible time in commuting on without ride-sharing detours.

Different algorithms can show optimums for different objective functions. The optimum obtained in terms of the maximal number of served clients (Function (3.5.1)) does not necessarily lead to the maximisation of the value of Function (3.5.5) defined in this subsection.

The following simplification of real-life taxi service is considered in this problem setting. The client does not need immediate report from the taxi service. The client is considered to be satisfied if it is taken within its constraints and considered to be not satisfied if any of its constraints is not respected. The client will quit the system at the end of its time window even if its taxi is already approaching.

3.6 Conclusion

In this chapter, the model and problem setting were provided. Firstly, we describe the electric aTaxi system.

Secondly, we provide the model of the described system. In the proposed model, all components and parameters are divided into 2 main categories: static and dynamic ones. Static elements are given

as input, whereas dynamic elements are changing over the simulation period. As an important part of the dynamic parameters, the client utilities can be highlighted. We provide here 2 different models. The first one provides deterministic client utilities, which are the classical representations of client time windows, walking and ride-sharing constraints. The second one provides stochastic nonlinear client utilities which represent better the random nature of human behaviour.

In the end, we set the electric aTaxi management optimisation problem. The problem setting includes 3 main subproblems: charging management, empty vehicle redistribution and ride-sharing problems.

The implementation of this model as well as the search of solutions of the optimisation problems can be provided using different techniques and methods. In the next chapters, the heuristics and methods for deterministic and stochastic client utilities are provided for the set of problems posed in this chapter.

Chapter 4

Functional architecture of aTaxi system management

4.1 Introduction

This chapter aims to provide a framework for the optimisation problem of the electric aTaxi system.

The formalisation of the model of the system, described in Chapter 3, allows us to move forward for the providing of its solution. The optimisation of aTaxi systems can be considered in online or offline contexts. The online version of the problem supposes that the demands arrive in real time and are not known in the beginning, whereas the offline version requires all the demands to be provided initially. In this work, we deal with the online context in which customer demands occur over time and have to be answered in real time, and where vehicles are available for ride-sharing and require electric recharging management. The offline context of the same problem is used only for the solution evaluation.

In this chapter, we reveal necessary modules for solutions of the main subproblems highlighted in Chapter 3 (Section 3.5), such as charging management, empty vehicle redistribution, and dynamic ride-sharing problems. However, the NP-complexity (see Chapter 2) of these subproblems requires to provide scaling in the considered aTaxi system.

In order to provide more evident and logical structure, this chapter is organised as follows. We start by presenting the proposed methods for zone decomposition in the system, based on the given environment. Section 4.2 provides methods of city road network graph partitioning and details a module for the graph covering based on this partitioning. Further, we discuss algorithmic solutions for allocated subproblems. The zone decomposition of the road network graph requires specific methods for different subproblems, which take this decomposition into account. In Section 4.3 we describe different patterns in charging stations demand and distributions, and charging strategies for autonomous vehicles, based on demand level and empty slot availability. In Section 4.4 we provide new station based methods of Empty Vehicle Redistribution for the problem without time windows. For the problem with time windows we provide their time limited modifications. In Section 4.5 we propose a new method of ride-sharing, which is based on zone and client indexing. Section 4.6 is dedicated to the algorithmic solution. We extend the discussed methods for EVR in order to provide more robust methods of zone-to-zone redistribution.

The conclusion in Section 4.7 summarises the relationships between the different methods discussed in this chapter, and provides areas of use and limitations of these methods.

4.2 Zone decomposition of road network graph

Graph clustering is used in a lot of different transportation approaches. In order to diminish computational complexity and memory requirements of the optimisation, firstly we have to divide the road network into specific clusters (zones), as described in Section 3.3. Therefore, we develop methods of

city graph partitioning [160] (see Subsection 2.2.5) and we provide a module for the graph covering based on this partitioning.

Partitioning of a road network graph has the following *constraints*:

- Each zone $z \in Z$ is a strongly connected subgraph of the road graph (connectivity constraint).
- Each vertex $v \in V$ of the main graph is included in exactly one zone (uniqueness constraint).
- The number of vertices in the zones is approximately the same (uniformity constraint).

Each defined zone has *properties* that define its fitness for the aTaxi management problem:

- A zone $z \in Z$ should have a small diameter.
- A zone $z \in Z$ should have a low sum of weights, which are represented by distances.
- A zone $z \in Z$ should have a big density, thus, the relation of the number of arcs $|A_z|$ to their maximal possible number.

We introduce the reduced graph in terms of zones as following. Reducing of G to G_r is defined as follows. Each $i-$th zone z_r is to be replaced by vertex $Vr_i \in V_r$, and the existence of arc from Vr_i to Vr_j is defined by existence of at least one arc from set of vertices belonging to $z_i \in Z$ to one of vertices belonging to $z_j \in Z$. Thus, zones z_i and z_j are adjacent if and only if it exists an arc between V_{r_i} and V_{r_j} in reduced graph.

We fix the number of zones in the system to be equal to k and maximise the fitness of their totality under the constraints described above. The problem of the graph partitioning hence reduces to the problem of minimising the following objective function

$$F = g(G_r) + \sum_{i=1}^{k} f(z_i), \tag{4.2.1}$$

where $f(z_i)$ is the fitness function of zone z_i, and $g(G_r)$ is the fitness function of reduced graph G_r.

The classical solutions of the clustering problem, such as k-medoids [282, 303], which has polynomial complexity, turned out to be impossible to apply due to the discussed above constraints imposed on the problem. In the following subsections, we provide new methods that we developed for solving of this problem.

4.2.1 Fill Clustering Method

The Fill Clustering Method (FCM) is developed based on the Differential Evolution (DE) method and aims to provide partitioning of the road network graph. It respects the constraints of connectivity, uniqueness, and uniformity and takes into account fitness properties. The first stage of FCM provides the greedy uniform partitioning, while the second one provides the redistribution of the vertices in order of fitness improvement.

The algorithm for this method is as follows:

1. The first stage of algorithm is creating clusters. An initial set of $|V|$ clusters is created, where each cluster consists of exactly one vertex. In next steps, these clusters will be merged.
2. The second stage of the algorithm is to obtain a set of clusters, possibly non-optimal, satisfying the condition of connectivity. This method is based on the greedy heuristic.
   (a) At each step of the algorithm, a cluster z_u is selected, consisting of the minimal number of vertices; if there are several of them, one is chosen randomly.
   (b) From clusters adjacent to z_u, the cluster with the minimal number of vertices z_v is selected and merged with the z_u cluster.
   (c) The second stage ends when the number of clusters decreases to k.
3. The third stage of the algorithm is to improve the target function in a greedy way, either by exchanging vertices in adjacent clusters, or by transferring vertices from one cluster to another. Let function $D(z_i)$ be the diameter of cluster z_i, $S(z_i)$ be the total sum of arc weights of cluster z_i, $C(z_i)$ be the total number of inner arcs of cluster z_i, and $F_C(G_r)$ be the function that measures the connectivity of graph G_r. Thus the general fitness function can be defined as:

$$F = C_4 F_C(G_r) + \sum_{i=1}^{k} \left(C_1 D(z_i) + C_2 S(z_i) + C_3 C(z_i) \right), \tag{4.2.2}$$

where C_i are weights of the relevant factors (constants for a particular execution of the algorithm). Then, a certain variant of the genetic algorithm is used, the purpose of which is to carry out a selection of the belonging of individuals that improves the state of the population. It acts as follows.

(a) Chooses a pair of adjacent (in the original graph) vertices u and v belonging to different clusters z_u and z_v.

(b) Calculates the value of the objective function for the following options:

- u is transferred to cluster z_v, and v is transferred to cluster z_u. If one of clusters z_u and z_v becomes disconnected, the objective function value is considered to be infinity.
- u is transferred to cluster z_v.
- v is transferred to cluster z_u.

The best option out of the four, including original setting, is to be chosen and the cluster structure is to be changed accordingly.

4. The third stage ends when the algorithm termination condition is reached, for example, if no progress is made within a certain number of steps.

The number of possible graph partitions into clusters is finite. The process of differential evolution in every step aims to reduce the value of the fitness function. A finite number of attempts is allocated per iteration of the DE method, and if the function does not decrease after this number of attempts, the DE process ends. This method does not necessarily lead to the global minimum of a function, but to the local one. Complexity of this method depends on number of iterations N_{iter} and on size of initial population N_{psize} and equals $N_{iter} \cdot N_{psize}$.

Coefficients C_i can vary for different test cases. To obtain them, the following training algorithm is proposed. Different values of C_i lead to partition of graph G to different clusters. Thus we can suppose that obtained result, in terms of converged fitness function F^*, can be represented as a function of the initial graph and the set of coefficients $F^* = \Phi(G, C_i)$. In this case function $\Phi(G, C_i)$ is a deterministic function, which returns a scalar generalised measure of the quality, and, under fixed G, depends only on coefficient set. Thus, the problem reduces to the search of the optimum for function $\Phi'(C_i)$, which is produced by the method of differential evolution.

4.2.2 Spill Clustering Method

Some of the algorithms and the methods proposed in this work rely on the overlapped clusters (zones). This structure is necessary especially because of the demand structure. The demand arrives to one of vertices of the road network graph, but the client can agree to walk a certain distance to reach another station in order to choose earlier departure.

Thus, the zone overlapping in the system is considered, so that the departure from some vertex will always mean that the client will be served in one of vertices of the same zone. The overlapping is characterised by geographical distance between vertices. **Algorithm** 1 for zone z returns an extended zone z_c. Recall that w^{max} represents the maximal walking distance accepted in the system.

Algorithm 1 Extension of zone.

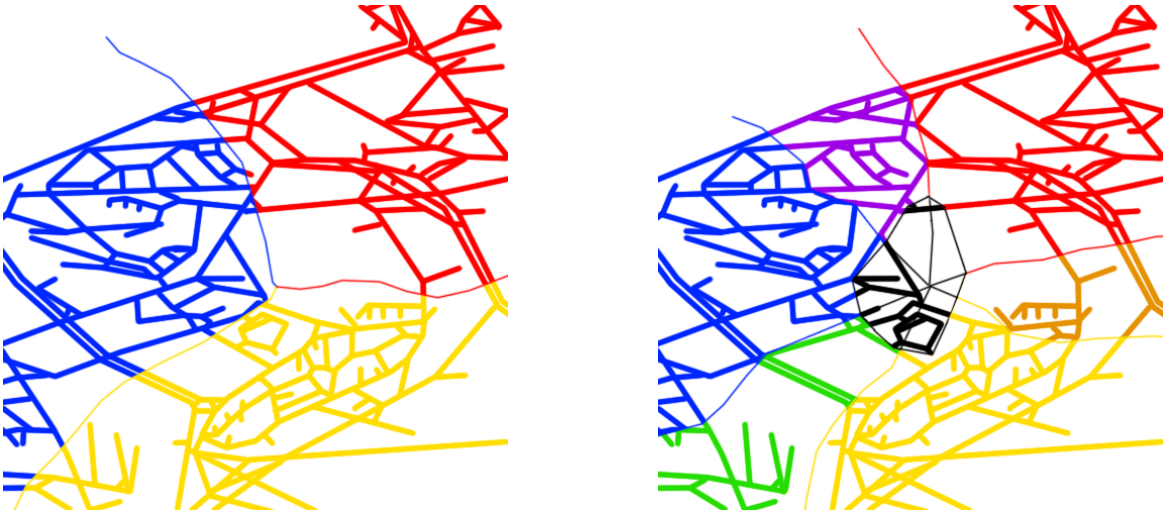
```
1: procedure ExtendedZone(z ∈ Z)
2:     z_c = z
3:     for nv ∈ V do
4:         for v ∈ z do
5:             if Distance(v, nv) < w^max then
6:                 z_c ← z_c ∪ nv
       return z_c
```

We propose the new Spill Clustering Method (SCM) for the covering, that saves fitness properties and connectivity constraint, but that does not satisfy the uniqueness constraint. The first step of the algorithm of this method is to execute FCM, then **Algorithm** 2 takes place.

Algorithm 2 Spill clustering algorithm.

```
1: procedure SpillClustering(g : Graph)
2:     Perform FillClustering(g)                    # calculate set of zones Z
3:     Z_c ← ∅                                      # setting the set of overlapped zones Z_C
4:     for z ∈ Z do
5:         z_c ← Extendedzone(z)
6:         Z_c ← Z_c ∪ z_c
       return Z_c
```

All potential zone assignments are computed during this algorithm. However, the final redistribution of zones and the rejection of uniqueness do not occur on initial set of zones, but on new set Z_c. This is provided in order to avoid multiple zone boundary shifts, which may also lead to their coincidence with the full network graph.



a) b)

Figure 4-1 – Transition from FCM to SCM output on part of Porto road network graph. a) FCM output, b) SCM output.

Summary

This Subsection introduces two new methods for clustering graphs, dividing them into zones. Traditional graph clustering methods prefer zones with approximately the same number of nodes in each zone (uniformity constraints). This approach does not accurately reflect the reality of the road network. The proposed algorithms are based on minimization of the objective function (4.2.1) and its version (4.2.2). The first method, FCM, uses a combination of greedy methods to obtain an initial guess and to refine the partition. The coefficients in the objective function (4.2.2) are calculated using learning algorithms.

The second method, SCM, does not use uniqueness constraints and allows to obtain a set of overlapping zones in which some stations belong to several zones at the same time.

These methods allow you to automate the process of obtaining and split the graph into zones more suitable for use in resource allocation problems similar to those solved in this work.

4.3 Charging management

In this section, we describe the charging system. It includes the available charging station prediction, their distribution and electric aTaxis charging strategies.

4.3.1 Station empty slot probabilities

The parking and charging stations in the system are predefined and can be used by other vehicles (for example, private) in the city. In the model, we consider, that for every station (see Subsection 3.3.2) the arrival rate is known, and depends on day type and period. The arrivals are modelled as Poisson processes. For every station γ the empty slot probability at moment t is defined as $P_{\gamma}^{available}(t)$ and depends on day type and period.

The estimation of empty parking place probability is calculated based on the current location of the vehicles and the current occupancy of the station, that can be found based on satellite data [146], wireless communication [51, 26], station statistics [223], magnetic sensors [75], or Poisson arrival based conditional probabilities [257].

Whenever the arrival rates to stations strictly depend on area, day, and day period, the departure rates from the stations show strong correlation with arrival rates. For common vehicles, the expected parking time shows 3 main patterns: night stays, working time stays and other stays. The data can be obtained easily using parking automates that derive the arrival and parking rates.

Let us describe the simplified model of probability estimation of empty parking slot based on its capacity and current occupancy of the station and arrival characteristics which we use in this work.

If S_{γ}^{max} is the maximal number of vehicles station γ can contain, we note $S_{\gamma}(t) \leqslant S_{\gamma}^{max}$ the current number of vehicles in the station. Thus the number of empty slots is $S_{\gamma}^{empty}(t) = S_{\gamma}^{max} - S_{\gamma}(t)$. Let us consider that the current moment is t_0 and that expected time to arrive to the station is τ.

Assuming that common vehicles arrive to station γ with the Poisson arrival rate ID_{γ} during time period T_i and leave the station with the Poisson departure rate OD_{γ} during time period T_o, the intensity of arrivals is $\lambda_{\gamma}^{i} = \frac{ID_{\gamma}}{T_i}$ and the intensity of the departures is $\lambda_{\gamma}^{o} = \frac{OD_{\gamma}}{T_o}$. A fundamental property of independent Poisson processes is that their pooled process is also a Poisson process with arrival-rate parameter equal to the sum of the individual arrival rates. Thus the intensity of the occupancy changement at the station is $\lambda_{\gamma} = \lambda_{\gamma}^{i} - \lambda_{\gamma}^{o}$, that can be also negative. Consequently, the expected number of arrivals during the time period $(t_0, t_0 + \tau)$ can be calculated as $\Lambda_{\gamma}(t_0, \tau) = \int_{t_0}^{t_0+\tau} \lambda_{\gamma}(t)dt$.

Thus, if the current time is t_0 and the expected arrival moment to the station is at $t_0 + \tau$, the probability, that the vehicle will find at least one empty slot when arriving to the station can be estimated as

$$\sum_{j=0}^{S_{\gamma}^{empty}(t_0)-1} \frac{\Lambda_{\gamma}(t_0,\tau)^j}{j!} \cdot e^{-\Lambda_{\gamma}(t_0,\tau)}, \lambda_{\gamma} > 0.$$

If $\lambda_{\gamma} \leqslant 0$, we estimate the probability to find an empty slot to be equal to 1.

When choosing a charging station, the vehicle chooses the nearest station with the probability to find an empty slot more than a predefined number $P_{constraint}^{available}$. If there are no empty slots (places) when arriving to the station, the vehicles from the modelled autonomous taxi system make a queue, that is considered to be a prioritised one.

The estimated waiting time of one slot release generally can be calculated for every station separately. Considering the Poisson arrivals and departures, the estimated time to release one slot in station γ can be calculated as $\frac{1}{\lambda_{\gamma}^{o}}$. In the model, we consider these numbers to be predefined for every station for every day and day period. For model simplification reasons, the input data for every station contain only the information about an empty slot probability at every moment and the information of necessary waiting time if there are no empty slots on arrival. As a general rule, the more is the probability to find an empty slot at the station, the less is potential time to wait. Thus the data can

be generated according to predefined regression rule. If prioritizing the autonomous vehicles in the queue, the waiting time for an empty slot at the station depends on the size of the station. Using the rough data that average parking time in public parkings equals to 1 hour, the waiting time $T_\gamma^{available}(t)$ as a function of the station empty slot probability $P_\gamma^{available}(t)$ in this work is assumed to be equal to $T_\gamma^{available}(t) = \dfrac{1 - P_\gamma^{available}(t)}{S_\gamma^{max}} \cdot 60$ minutes.

4.3.2 Station location distributions

We assume that both charging and parking station locations are known at the start of the simulation. If the existing data provide all necessary coordinates, the simulator will use them as given ones. Otherwise, in order to model the locations of stations, we use the following method.

Given the number of charging and parking stations, the separate creation of stations of both types is to be made based on the following city clustering method. The road network is to be divided into regions, minimizing the total sum of distances from all the vertices to their nearest station. We use firstly Fill Clustering Method (Subsection 4.2.1), and after we define the station coordinate using k-medoids method [113].

In order to provide mode robust solution, the statistics of the client demand for the available time period are used to provide a weighted k-medoids problem [282, 303]. The proposed method minimises the sum of distances from all the clients, both from origin and destination points, that can be start of end points of the vehicle trajectory after or before the charging cycle.

4.3.3 Charging strategies

The vehicles in the system have a finite battery, thus, the vehicle should avoid the possibility to become discharged (especially when it has passengers inside).

We propose a new method of vehicle charging. Every vehicle has the following strategy, defined by specific rules when it starts to move to the nearest available charging station to start the charging process. We divide these rules into three categories.

1. *Strong rule.* Before accepting any ride or any change in the current one, the vehicle checks if its current charge is sufficient to carry out current trip sequence with an additional ride, and to be able afterwards to reach the charging station with empty slot with high probability. If there are no empty slots in the station, the vehicle will park and wait for the possibility to charge.
2. *Preferable rule.* If the vehicle current charge level is less than 30 percent, then if there are no clients to take at its current station, the vehicle moves to charging station.
3. *No demand rule.* At any charging level, if there is no demand in the whole system, some vehicles (based on the predicted demand) move to charging or parking stations.

These rules are to be checked in the order listed above. Their totality guarantees of the continuous system functioning. Here, the strong rule guarantees that the vehicle will not be discharged during the ride. Preferable rule ensures that vehicles in the system will have a sufficient level of charge at every moment. 30 percent is the first of standard charge level warnings [101]. Finally, no demand rule ensures the diminishing of useless empty trips (thus, trips without specific goals).

When the vehicle is charging, it interrupts its charging process according to the following rules. These rules aim to free the charging slots and/or vehicles for the use.

1. *Strong rule.* If any vehicle has 100% charge level, it returns to the system.
2. *Demand rule.* Let the current demand in the system to be N clients and the current number of available vehicles to be N_V. Then, if $N > N_V$ and there is a set of vehicles that reached $\min(\frac{N_V}{N} \cdot 50\%, 30\%)$ charge level, one of the vehicles returns to the system and gets available.

We use both rules together to guarantee the functioning of the system. Strong rule is necessary to free the unused charging slots. If any guarantee has 100% charge level, it returns to the system. Demand rule aims to make vehicles which are charged enough, return to the system. The 30 percent borderline refers to the standard charge level warning. The percentage of $\frac{N_V}{N} \cdot 50$ signifies the necessity

to free more vehicles in case of bigger demand. The 50 percent borderline is a value that is chosen based on simulations on test networks.

In this work we use the random combination of two main heuristics for choosing and empty and not booked, that are the following.

We return to the system from parking or charging station either the vehicle with minimal travel time to one of unserved demands or the vehicle with maximal charge level. This aims to ensure at the same time the sufficient charge level of available vehicles and the minimisation of the empty runs.

4.4 Empty vehicle redistribution methods

In this section, we develop two new methods for empty vehicle redistribution. The first is reactive method and the second is a proactive one (see Subsection 2.3.1).

The new reactive algorithm *"Send The Nearest"* on greedy matchings in bipartite graph. This graph is constructed by connecting waiting clients and available vehicles. This EVR is based on the nearest pairs between the available vehicles and waiting clients. The stations with waiting clients are to be ranged based on the time from the nearest available vehicle to the station.

The new proactive (redistribution) algorithm *"Index-Based Redistribution (IBR)"* is based on maximal station index. The idea of the station index is related to the "virtual station status" described by Kek et al. [153], but we use a different method for index calculation (authors focus on the number of vehicles and do not consider the client waiting times). This method is described below.

4.4.1 Empty vehicle redistribution without time windows

Index-Based Redistribution In this paragraph we propose a novel Index-Based Redistribution algorithm that incorporates the expected client waiting time into a numeric index, thereby making it responsive to both expected waiting times as well as (expected) length of client queues.

The client arrival process is assumed to be a stationary Poisson point process. The demand data provide the client arrival predicted rates N_1, ..., N_i during time periods T_1, ..., T_i, respectively.

During each time period $(t, t+\Delta t)$ with expectancy of the number of arriving clients $N(t)$, the intensity of the flow is defined as $\lambda(t) = \frac{N(t)}{\Delta t}$, and therefore the expected number of arrivals during time period $(t_0, t_0+\tau)$ is $\Lambda(t_0, \tau) = \int_{t_0}^{t_0+\tau} \lambda(t)dt$.

We define the *probability of at least one client arrival* during time period $(t_1, t_1+\tau)$ as $P_{arr>0}(t_1, t_1+\tau) = 1 - e^{-\Lambda(t_1,\tau)}$.

Under both deterministic and stochastic client preferences (see Subsection 3.4.6), the client's acceptable waiting time is upper-bounded by some value, and can be different for different clients. Also, client preference for waiting time is non-linear [130]. To take this fact into account, and to allow calculation of potential client preference heuristics (for example, to introduce risk aversion), a (dis-) utility $u_p(t_p^{dep} - t_p^0)$ as function of client waiting time is introduced.

For every station we introduce the *"station index"*, which is calculated as a measure of expected maximum client disutility at the time of pick-up by the nearest vehicle, including the time it takes for the vehicle to arrive at the station. The station index calculation is based on the number of already arrived clients and on "virtual" clients that will arrive in future.

For every station $s \in V$, we determine nearest available vehicle $\xi_s^{nearest}$, given its trajectory. Given the current time t_1, we calculate that $\xi_s^{nearest}$ will arrive to the station at the time t_2.

For every station s, the "client surplus" (CS_s) is calculated as the total number of clients currently at this station minus the total number of vehicles moving to this station to take clients.

Suppose the current time from the start of simulation is t_1 and it was calculated that the nearest vehicle will arrive to the station at time t_2. Thus, for station s, the "station index" I_s is calculated as follows.

If $CS_s > 0$ (vehicle deficit): Let the maximum waiting time among the clients at station s be $T_s = \max_{p \in P_s} t_p$, where t_p is the current waiting time of client p. The station index I_s is defined as

$u(t_2 - t_1 + T_s)$. In other words, the station index is defined as the disutility of the longest waiting client at the station, at the time of its predicted departure.

If $CS_s = 0$ (balanced): The station index I_s is equal to the maximal expected disutility of the first arriving client:

$$I_s = \max_{t_1 < \tau < t_2} [P_{s,arr>0}(t_1, \tau) u(t_2 - \tau)].$$

If $CS_s = -X$ (surplus of X vehicles): The time expectancy $T_{s,X}$ of X clients arriving is calculated from the given type of demand distribution. If $t_1 + T_{s,X} \geqslant t_2$, station index $I_s = 0$, otherwise it is calculated as in $CS_s = 0$ (balanced) case, changing the start time from t_1 to $t_1 + T_{s,X}$:

$$I_s = \max_{t_1 + T_{s,X} < \tau < t_2} [P_{s,arr>0}(t_1 + T_{s,X}, \tau) u(t_2 - \tau)].$$

In other words, in case of non-positive client surplus, the index is equal to the probable disutility of the first arriving client at the time of its departure.

To generalize the three cases above, the station index is a measure of expected maximum client disutility at the time of pick-up, including the time it takes for the nearest vehicle to arrive at the station.

The station index calculation is described in details in **Algorithm** 3.

Algorithm 3 Station index calculation algorithm.

```
1:  procedure StationIndex(s : Station)
2:      input: station to compute the index
3:      output: calculated station index
4:      CS ← NumberOfArriving − NumberVehiclesToTake
5:      t1 ← CurrentTime
6:      t2 ← NearestVehicleArrivalTime
7:      if CS > 0 then                                   # Vehicles deficit
8:          Ts ← MaximalWaitingTimeAmongsClients
9:          Is ← u(t2 − t1 + Ts)                         # u(t) — utility function
10:     else if CS = 0 then                              # Balanced case
11:         Is ← max_{t1<τ<t2}[P_{s,arr>0}(t1, τ)u(t2 − τ)]
12:     else if CS < 0 then                              # Vehicles surplus
13:         X ← −CS
14:         if t1 + T_{s,X} ⩾ t2 then                    # T_{s,X} — Expected time of X clients arrival
15:             Is ← 0
16:         else
17:             Is = max_{t1+T_{s,X}<τ<t1}[P_{s,arr>0}(t1 + T_{s,X}, τ)u(t2 − τ)]
18:     return Is
```

Index-Based Redistribution algorithm

The Index-Based Redistribution is based on the station indexes described above.

Given the station s with maximal station index I_s, if there is an available vehicle $\xi_s^{nearest}$ at station s, assign it to the longest waiting client if any. If there are no waiting clients (yet), reserve $\xi_s^{nearest}$ for the next arriving client.

If there is no available vehicle at station s, vehicle $\xi_s^{nearest}$ will be chosen as nearest available vehicle at other stations or moving in the network. There are three possible cases.

- If vehicle $\xi_s^{nearest}$ is idle at its station, this vehicle will move to station s to serve a client.

- If vehicle $\xi_s^{nearest}$ is moving and its destination is station s, then $\xi_s^{nearest}$ is reserved for serving the longest waiting client immediately after reaching s.
- If vehicle $\xi_s^{nearest}$ is now in a trip to another station, it is booked to make its next trip to station s.

The Index-Based Redistribution algorithm is described in pseudocode in **Algorithm** 4.

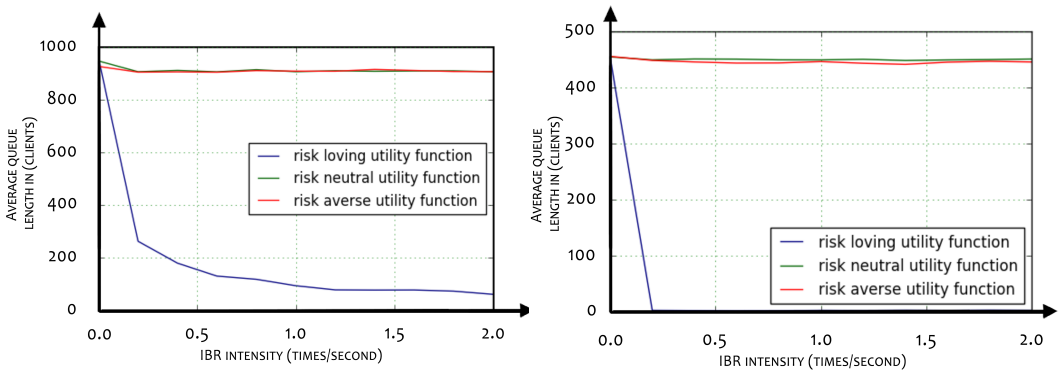
Algorithm 4 Index-Based Redistribution algorithm.

```
1:  procedure IndexRedistribution(set of Station SS)
2:      input: set of station
3:      output: perform action for best possible vehicle ξ
4:      s : Station ∈ SS with maximal StationIndex(s)
5:      Vehicle ξ ← FindAvailableVehicleInStation(s)
6:      if ξ ≠ ∅ then                          # There is an available vehicle in the station
7:          if s.pqueue.size > 0 then          # There are clients in the station
8:              client p ← s.pqueue.popfront   # Take longest waiting client from
                                               # queue
9:              SendToDestination(p)           # Dispatch vehicle
10:         else
11:             Reserve(ξ)                     # Reserve the vehicle for next arriving client
12:     else                                   # There is no empty vehicle in the station
13:         Vehicle ξ ← FindNearestAvailableVehicle
14:         if ξ.idle then
15:             SetCurrentTrip(ξ, s)           # Send the idle empty vehicle to this station
16:             SetNextTrip(ξ, p.destination)  # Reserve the vehicle for next trip
                                               # with client
17:         else if v.destination = s  then    # if already moving to this station
18:             SetNextTrip(ξ, p.destination)  # Reserve the vehicle for next trip
                                               # with client
19:         else                               # The vehicle is moving to another station
20:             SetNextTrip(ξ, s)              # Book empty trip to station s to
                                               # pick up client p
```

Function $FindNearestAvailableVehicle$ can be calculated in terms of shortest distance or travel time. In the remainder of this thesis it is based on travel time.

Utility function

There are many possible formulations for the (dis-)utility function to be used in the indexing algorithm, convex and not-convex, such as quadratic, cubic, exponential, tangent, linear, etc. Different formulations can be used to make the heuristic more or less risk averse. We compare three alternative formulations using the pure IBR strategy. The comparison is made on a small subnetwork of the Saclay network (France), using two different demand levels. As risk-loving (or aggressive) function we chose $u(t) = exp(t) - 1$, since it aggressively penalises long waiting times, as risk-neutral $u(t) = t$, and as risk-averse (least aggressive) $u(t) = \sqrt{t}$. Figure 4-2a shows the results for rush-hour demand and Figure 4-2b the results for half of the demand. Both figures clearly show that only the risk-loving (aggressive) function shows good performance, decreasing the average queue length with increased intensity of the IBR algorithm. For the remainder of this thesis we choose therefore when using IBR the risk-loving function $u(t) = exp(t) - 1$.



(a) (b)

Figure 4-2 – Comparison between different utility functions: (a) rush hour demand, (b) off-peak demand.

Note that the utility function in this algorithm differs with the utility function discussed in Subsection 3.4.6. Here, the utilities were introduced to implement aggressivity to EVR strategies.

Heuristic Nearest Neighbours algorithm (HNN) The HNN algorithm, discussed in Subsection 2.3.1, can be described as highly simplified IBR, without taking into account the expected client arrivals and with the linear utility function $u(t) = t$. Thus, the HNN redistribution can be described as the IBR when changing index I_s to T_s^m which is the maximal waiting time of clients at station s at the arrival moment of the nearest vehicle.

4.4.2 Empty vehicle redistribution with time windows

The EVR problem with time windows differs from the classical EVR problem in one main aspect considered in this chapter: for every client there is a predefined time window, and when the client waiting time reaches its time window limit, he/she will quit the system. Thus, in the optimisation problem, there may not be a reason to send a vehicle to the client that will quit the system before the vehicle arrives to the station.

Taking into account these considerations of client's time windows, the IBR as well as the HNN algorithms were updated to not consider in the optimisation problem clients that cannot be served within their time windows. We call the updated algorithms Index-Based Redistribution Time Limited (IBRTL) and Heuristic Nearest Neighbours Time Limited (HNNTL).

Heuristic Nearest Neighbours Time Limited algorithm (HNNTL) The calculation of the maximal waiting time at the moment of the nearest vehicle arrival T_s^m to station $s \in V$ for the HNNTL method is changed as follows: for every station s in the system, the vehicle $\xi_s^{nearest}$ is found. Suppose that the current time from the start of simulation is t_1 and calculated nearest vehicle will arrive to station s at the time t_2.

Thus, the value T_s^m can be calculated as follows:

$$T_s^m = \max_{p \text{ at } s, t_2 \leqslant t_p^1} (t_2 - t_p^0).$$

It represents the maximal waiting time within all the clients at station s within their time window at the moment of the nearest vehicle arrival. The T_s^m calculation is described in details in **Algorithm** 5.

Index-Based Redistribution Time Limited algorithm (IBRTL) This method is based on IBR method, but the calculation of the station $s \in V$ index I_s is changed as follows. For every station, only clients, such that the current time of the client plus the time needed to reach this station by the

Algorithm 5 Calculation of the station with maximal waiting client for the HNNTL.

procedure StationMaximalTL(s : *Station*)
  input: station to perform computation
  output: computed maximal waiting time for station
  $ClientsToConsider \leftarrow ClientsOnStation$ # *Copy clients from station*
  $t_1 \leftarrow CurrentTime$
  $t_2 \leftarrow NearestVehicleArrivalTime$
  for $p \in ClientsToConsider$ **do**
    if $t_2 > t_p^1$ **then** delete p from $ClientsToConsider$ # *Delete long waiting clients*
  $T_s \leftarrow MaximalWaitingTimeAmongClientsToConsider$
  $T_s^m \leftarrow T_s + t_2 - t_1$
  return T_s^m

nearest vehicle is less or equal to the time limit, are considered. So, the current list of clients in these stations is updated to not take into account clients that cannot be served within their time window. The index I_s calculation is described in details in **Algorithm** 6.

Algorithm 6 Calculation of station index for the IBRTL.

procedure StationIndexTL(s : *Station*)
  input: station to perform computation
  output: computed index for IBR
  $ClientsToConsider \leftarrow ClientsOnStation$
  $t_1 \leftarrow CurrentTime$
  $t_2 \leftarrow NearestVehicleArrivalTime$
  for $p \in ClientsToConsider$ **do**
    if $t_2 > t_p^1$ **then** delete p from $ClientsToConsider$
  $CS \leftarrow NumberOfClientsToConsider - NumberVehiclesToTake$
  if $CS > 0$ **then** # *Vehicle deficit*
    $T \leftarrow MaximalWaitingTimeAmongClientsToConsider$
    $I_s \leftarrow u(t_2 = t_1 + T)$ # $u(t)$ — utility function
  else if $CS = 0$ **then**
    $I_s \leftarrow \max_{t_1<\tau<t_2}[P_{s,arr>0}(t_1, \tau)u(t_2 - \tau)]$
  else if $CS < 0$ **then** # *Vehicle surplus*
    $X \leftarrow -CS$
    if $t_1 + T_{s,X} \geq t_2$ **then** # $T_{s,X}$ — Expected time of X clients arrival
      $I_s \leftarrow 0$
    else
      $I_s = \max_{t_1+T_{s,X}<\tau<t_2}[P_{s,arr>0}(t_1 + T_{s,X}, \tau)u(t_2 - \tau)]$
  return I_s

The comparison of the IBR and HNN methods with IBRTL and HNNTL is provided in the case studies in Chapter 7. The common IBR and HNN methods can be obtained from IBRTL and HNNTL when considering the infinite time windows, so the values of the objective functions of these methods will differ.

4.4.3 Comparison of empty vehicle redistribution methods

In this subsection we evaluate different redistribution algorithms mathematically on a simple linear network, to see if any of these methods outperforms the others in terms of average and maximum waiting times and the number of satisfied clients.

Let us consider a single line network with two stations and two vehicles (Figure 4-3). Each station has one client waiting for a vehicle. The driving time between the two stations is 5 minutes.

For such a network, the HNN method shows the same results as the IBR method, because this case study does not consider predicted clients. The Surplus/Deficit redistribution can be applied only when vehicles are located at the stations, so this method is not applicable to this case study.

SNN, IBR, and STN algotithms were evaluated and compared to see which algorithm produces the best maximum waiting time, the best average waiting time, and the biggest number of satisfied clients.

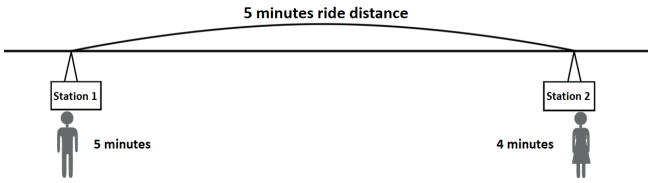


Figure 4-3 – The simple line network.

We will now consider four cases with different locations of two vehicles:

Case 1. The first vehicle is located 3 minutes ride time from Station 1 and the second vehicle is located between Station 1 and Station 2, with a ride time of 1 minute to Station 1 and 4 minutes to Station 2.

Case 2. The second vehicle is located 7 minutes ride time from Station 1 and the first vehicle is located 1 minute ride time from Station 1.

Case 3. The first vehicle is located between Station 1 and Station 2, with a ride time of 1 minute to Station 1 and 4 minutes to Station 2. The second vehicle is located 5 minutes ride time from Station 2.

Case 4. The first vehicle is located between Station 1 and Station 2, with a ride time of 4 minutes to Station 1 and 1 minute to Station 2. The second vehicle is located 2 minutes ride time from Station 2.

As shown in Table 4.1, there is no single best algorithm which suits all evaluated cases. For example, the IBR algorithm is the best in cases 1 and 4, the SNN algorithm in cases 3 and 4 and STN algorithm in cases 2 and 3. This demonstrates that even in a very simple evaluation case, different situations require different algorithms. We therefore conclude that a mix of algorithms may provide a more robust strategy and less sensitivity to particular demand characteristics and vehicle distributions in the network. Moreover, we note that the greedy type of assigning vehicle-client pairs sequentially often leads to suboptimal solutions.

Table 4.1 – Average and maximal waiting times, in minutes.

	Case 1		Case 2		Case 3		Case 4	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
IBR	**8**	**8**	13.5	16	11.5	15	**7.5**	**9**
SNN	9	12	13.5	16	**7.5**	**9**	**7.5**	**9**
STN	9	12	**8.5**	**12**	**7.5**	**9**	8.5	12

As shown in Table 4.2, for the problem with time windows, there is also no single best algorithm which suits all evaluated cases — in every case, the number of clients not served within their time windows (considered to be 10 minutes) reaches its minimum using either IBR, SNN, or STN algorithms.

When combining these algorithms, in every optimisation step, the system chooses the redistribution method as well as the number of available vehicles to redistribute. It is thus not necessary, that the system redistribute all the available vehicles.

The selection of redistribution algorithm is based on random draws, given a predefined intensity I_i for each algorithm, which is the number of times that algorithm is expected to be called per optimisation time interval. For example, let the IBR intensity be I_1, and the SNN vehicle redistribution intensity

Table 4.2 – Number of not served clients for the problem with time windows.

	Case 1	Case 2	Case 3	Case 4
IBR	**0**	2	1	**0**
SNN	1	2	**0**	**0**
STN	1	**1**	**0**	1

be I_2. Then for each optimisation interval there are $I = I_1 + I_2$ random selections of redistribution algorithms with probabilities $p_1 = \frac{I_1}{I}$ and $p_2 = \frac{I_2}{I}$, respectively.

Each vehicle selection is independent and is based on attributes of stations, vehicles and demands with parameters which can be set externally. These strategies can be combined based on instantaneous conditions and/or time intervals, for example controlled by current demand in the system.

The matching problem

In all proposed methods the sending depends on core parameter, which can be station index, number of clients, maximal waiting time of the clients, or the minimum distance to an available vehicle with the presence of clients at a given station. The mixed methods aim to send vehicles in greedy basis. After determining for each station the index which depends on chosen redistribution strategy, we can obtain the set of vehicles available for redistribution. The greedy methods based on step-by-step matching between the first-sorted elements can be considered, for example, to send the nearest available vehicle to the station with the largest index. This method can show good convergence, but it is unlikely to always reach the optimal solution. In addition it may cause unnecessary runs of empty vehicles.

To improve the solution we represent the data of the 2 sets in the form of a bipartite graph, where the edges are the travel times from the available vehicles to the stations that need to be served.

Let it be required to find matching vehicles to the N top stations in the network. To achieve a minimum average waiting time for clients, it is required to find matchings minimising the total weight sum of the used edges. In general (if we consider weights to depend on the number of passing vehicles), this problem is NP-hard, but for some specific cases, when the weights are constant, polynomial solutions exist. For example, if the number of serving stations is equal to the number of available vehicles, the Hungarian algorithm [159] can be used. In the following we investigate some heuristic methods of the matching problem solution.

In every call of the matching function in the network, based on the statistics about the stations, we will serve the top N stations basing on the chosen parameter. The number of matchings can never exceed the number of available vehicles.

We use the following example to show that a greedy algorithm may not provide the optimal solution. Let us choose the SNN strategy based on maximal client waiting time.

Figure 4-4 shows a simple network with 2 stations and 2 available vehicles that can be represented as the bipartite graph in Figure 4-5.

Figure 4-4 – The sample network.

The greedy solution can be used in two different ways: at each step we can choose a client with the maximal waiting time or we can choose the vehicle with the shortest path to any of the waiting clients. In the first case, the longest waiting client is in Station 1, so the nearest vehicle (2) will serve it. The next longest waiting client in Station 2 will then be served by Vehicle 1. The total waiting time is $(5 + 2) + (4 + 8) = 19$ minutes and the total run of empty vehicles is $8 + 2 = 10$ minutes. In

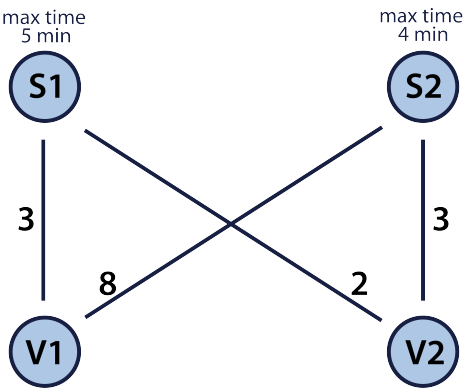


Figure 4-5 – The bipartite graph.

the second case, the second vehicle will be chosen for serving a client in Station 1, because it has the shortest time to reach the nearest client. In the next step the first vehicle serves a client in Station 2. Both scenarios lead to the same result.

The optimal solution would assign Vehicle 2 to Station 2, and Vehicle 1 to Station 1. The waiting times at departure departure time will be 8 minutes and 7 minutes for Station 1 and Station 2, respectively. Thus, both the average waiting time and the maximal waiting time will be smaller.

The Greedy+Hungarian matching algorithm

In order to diminish the complexity of this problem, we propose the following 2-steps method.

In the first step we call the greedy strategy for pre-assignment of the available vehicles to the station. In the second step the set obtained with N stations and N vehicles will be the input of the Hungarian algorithm.

The computational complexity of this complex method is estimated as follows. Let the number of requests be N_R and the number of free vehicles be N_C. The first step requires two sorting operations, thus its complexity is $T_1 = O(N_R \log N_R + N_C \log N_C)$. Let $N_H = \min(N_R, N_C)$. The second step requires $T_2 = O(N_H^3)$ which is the final computational complexity of the algorithm, because $T_2 \gg T_1$. This is polynomial hard.

4.5 Ride-sharing methods

This section investigates a new heuristic method of dynamic ride-sharing in road networks (see Subsection 2.3.2)

The proposed methods have to answer two main questions: which vehicle to assign to which customer and how to provide a reliable ride-sharing? These two problems are NP hard and only heuristic-based solutions are possible. Therefore, we propose an heuristic based on city graph clustering. The basic idea of this strategy is to identify the customers whose requests can be grouped and assigned to the same vehicle while guaranteeing a certain quality of service.

From the literature, we consider Searching On The Way (SOTW) and Searching in The Area of the origin and/or destination (SOTA) heuristics. Both methods have, as described in Subsection 2.3.2, different advantages and disadvantages. We use them both as benchmarks and as modules for reinforcement learning methods.

In order to deal with high computational complexity, we consider an approach which incorporates the proposed method in Section 4.2 (zone decomposition).

In this approach we consider a weighted graph G with the set of vertices V and the set of arcs A. P_i describes the set of clients on vertex v_i.

Let $SP(v_O, v_D)$ be the shortest path length (time) from $v_O \in V$ to $v_D \in V$. This can be calculated in time complexity of $O(Diam)$, where $Diam$ is the maximal distance in G in terms of number of

passed arcs.

Precalculation on network This first step consists of the following preprocessing:

1. **Partitioning** to $|Z|$ zones with $|z \in Z| \approx X = \dfrac{|V|}{|Z|}$ vertices in zone $z \in Z$ using FCM (see Section 4.2). This step is performed just once and has complexity $O(|V|^2)$.

2. **Precalculation 1:** Use Floyd-Warshall algorithm to build matrices $Dist[|V|][|V|]$ and $Best[|V|][|V|]$, where $Dist[u][v]$ is the shortest path from vertex $u \in V$ to vertex $v \in V$ and $Best[u][v]$ is the first vertex after u on the shortest path from u to v. Computational complexity of this step is $O(|Z|^3)$.

3. **Precalculation 2:** For each pair $(u, v), u \in z_u, v \in z_v, z_u \neq z_v$ and each zone $z_t \in Z$ calculate the number of shortest paths $SP_{u,v,t}$, $u \to v$, containing zone z_t. The number of vertices u in origin is $O(X)$ and in destination is $O(X)$. Let L be characteristic path length in graph G [123]. The complexity of finding if single vertex x is on the best path from u to v is $O(L)$, so total complexity for one pair z_u, z_v is $O\left(X^2 \cdot L\right)$. The number of (z_u, z_v) pairs, when $z_u \neq z_v$, is $O(|Z|^2)$. The total time complexity for all (z_u, z_v) pairs is $O(|Z|^2 \cdot L)$ and the memory complexity is $O(|Z|^2)$ for all best paths (when using best neighbours table).

4. **Reduced graph building:** R_{z_O,z_D} is the condensed reduced Direct Acyclic Graph without weighted vertices $W_t = \sum_{u \in z_O, v \in z_D} P_{u,v,t}$ and zero weighted arcs. The complexity of building this graph by DFS algorithm is $O(|V| + |A|)$.

5. **Weighted paths cost matrix building.** Build the 3-dimensional matrix W, where $W_{i,j,k}$ is shortest length of path from zone z_i to zone z_j through transit zone z_k. The memory and computational complexities of this step is $O(|Z|^3)$.

Main algorithm The next algorithm finds for a given vehicle ξ a set of potential clients to ride-share (RS set) and the main lines of the vehicle trajectories. Let the current location of vehicle ξ be $v_O \in V$ and the current trajectory is from v_O to $v_{D_1}, \ldots, v_{D_k} \in V$.

- For each zone $z_X \in R_{z_O,z_{D_1}}$ do:
  1. Let number of clients in zone z_X be $|P_{z_x}|$.
  2. Let the initial set of client candidates be empty: $Cand_X \leftarrow \emptyset$.
  3. For each client p_C in zone z_X check its candidature. Let z_{dest} be the zone of its destination.
  4. If z_X is in $R(z_X, z_{D_1}) \cup R(z_{D_1}, z_{D_2}) \cup \cdots \cup R(z_{D_{k-1}}, z_{D_k})$ or $z_{D_1} \cup z_{D_2} \cup \cdots \cup z_{D_k}$ is in $R(z_X, z_{dest})$ then add this client to the set of primary candidates $Cand_X \leftarrow Cand_X \cup p_C$. This operation has complexity $O(|V|)$ for each client, thus a total $O(|V| \cdot |C_{z_x}|)$.
- Let graph $R'_{z_O,z_{D_1}}$ be a copy of $R_{z_O,z_{D_1}}$.
- For each node in $R'_{z_O,z_{D_1}}$, $W'_X \leftarrow W_X \times |Cand_X|$.
- Calculate the maximal path weight $RWeight(z_O, z_{D_1})$ from z_O to z_{D_1}. The total complexity of the operation using matrix W is $O(|z_O| \cdot |z_{D_1}|)$.
- Let the start list of the sharing candidates be empty: $RS \leftarrow \emptyset$.
- For each zone $z \in RWeight_{z_O,z_{D_1}}$, $RS \leftarrow RS \cup Cand_z$. The complexity of this operation is $O(L)$.

The algorithm produces a matrix of maximal zone-paths between zones $RWeight_{z_O,z_{D_1}}$ and the list of the potential clients RH that can share their trips within the current vehicle trajectory.

It remains to choose the clients themselves. For achieving this, the greedy indexing algorithm is used. The input parameters for this algorithm are the following: the starting vertex v_O; the current trajectory which is from $v_O = v_{D_0} \in V$ to $v_{D_1} \in V$ and then to $v_{D_2}, \ldots, v_{D_k} \in V$; the list of zones $RWeight_{z_O,z_{D_1}}$ to pass when travelling from v_O to v_{D_1}.

Starting from the first zone to pass (the starting zone), for each client $p_C \in RS$ we calculate its ride-sharing index I_C as follows.

Given that the origin station of the client p_C is v_{O_C} and the destination station is v_{D_C} in zone $z_{D_C} \in R(z_{D_i}, z_{D_{i+1}})$, where $z_{D_0} = z_O$, we have:

- if $z_{D_C} \in R(z_{D_0}, z_{D_1})$,

$$I_C = \frac{SP(v_O, v_{O_C}) + SP(v_{O_C}, v_{D_C}) + SP(v_{D_C}, v_{D_1})}{SP(v_O, v_{D_1})};$$

- if $z_{D_C} \in R(z_{D_i}, z_{D_{i+1}})$, where $i \neq 0$,

$$I_C = \frac{SP(v_O, v_{O_C}) + SP(v_{O_C}, v_{D_1}) + SP(v_{D_i}, v_{D_C}) + SP(v_{D_C}, v_{D_{i+1}})}{SP(v_O, v_{D_1}) + SP(v_{D_i}, v_{D_{i+1}})};$$

- otherwise,

$$I_C = \frac{SP(v_O, v_{O_C}) + SP(v_{O_C}, v_{D_1}) + SP(v_{D_{k-1}}, v_{D_k}) + SP(v_{D_k}, v_{D_C})}{SP(v_O, v_{D_1}) + SP(v_{D_{k-1}}, v_{D_C})}.$$

For ride-sharing purpose, the system chooses on greedy basis the clients with the minimal indices not exceeding predefined constant $ratio_{max}$, that is the maximal delay, that the vehicle can make.

Summary

In this section we proposed a new algorithm, which aims to provide faster ride-sharing, is based on road network partitioning. The heuristic finds, for a given vehicle, a set of potential clients to ride-share and the main lines (passing zones) of the vehicle trajectory. Then, using these data, we provide the indexing algorithm in order to predict potential delay for clients. For the ride-sharing purpose, the system chooses on greedy basis the clients with the minimal indices.

The total computational complexity and memory consumptions of both static (precalculation) and dynamic (online) parts of the method provided in this section are polynomial in terms of demand and size of the system. The heuristics of searching for clients to ride-share have theoretical capacity to outperform the reference methods. A more detailed analysis of the method is provided in evaluation Chapter 7.

4.6 Zone-based Empty Vehicle Redistribution

EVR from station to station, as it is shown by both theorists and practitioners, can significantly improve the system performance from client perspective, and in terms of fleet-size. However, in real-life scenarii, especially in the case of asymmetry of origin-destination matrix due to work traffic, the EVR between different areas (zones) of the city is more actual. For example, during evening rush hours the main passenger flow is from office area to the residencies. In such a traffic, the sending of vehicles to the office area is preferable for the system performance purposes. The following example can illustrate this case.

Let us consider a simple line network (Figure 4-6) with 4 stations. Let at Stations 1, 2, 3 be 1 client willing to reach Station 4. Let at Station 4 be 2 clients willing to reach different stations, not marked in this subnetwork. Serving clients from Stations 1, 2 and 3 not only helps to serve more clients, but would also help to provide shared vehicle trajectory. In this section we provide several heuristical redistribution methods for zone-based demand and supply patterns.

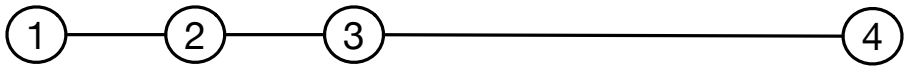


Figure 4-6 – Network for zone-based EVR example.

For proposed below original methods, the vehicle sending is to be made to the nearest station of the destination zone.

4.6.1 Surplus/deficit zone-based EVR

The empty vehicle redistribution in this case is carried out based on the demand of the clients and the supply of available vehicles. This method is inspired from classics in proactive EVR.

For each zone $z \in Z$, the number of available vehicles can be calculated and is referred to as $|\Xi_z^{avail}|$. The number of clients that are not served yet and have no assigned vehicle is referred to as $|P_z^{demand}|$. The zone surplus can be calculated as follows:

$$ZS_z = |\xi_z^{avail}| - |P_z^{demand}|. \tag{4.6.1}$$

The sending of an empty vehicle is carried out from the zone with the maximum surplus of empty vehicles to the zone with the maximum deficit, disregarding the distance between the zones.

4.6.2 Zone Index-Based EVR (IBR)

We develop here a zone-based modification of IBR method. In this case, the EVR is carried out on the basis of the zone index, which can take into account possible future arrivals. For every zone it is calculated as a measure of expected queue at the time of the nearest vehicle arrival, including the time it takes for the vehicle to arrive at the station. The zone index calculation is based on clients already arrived and on "virtual" clients that will arrive during certain time periods.

For each zone $z \in Z$ we determine the nearest vehicle $\xi_z^{nearest}$. Suppose the current time from the start of simulation is t_1 and it is calculated that the nearest vehicle can arrive to the zone at time t_2. Thus, for zone z, the "zone index" I_z is calculated as follows.

Let the sum of waiting times among clients of zone z be $T_z = \sum_{p \in P_z^{demand}} t_p$, where t_p is current waiting time of client p.

For predicted clients for every station we calculate the predicted sum of waiting times at the moment of vehicle arrival to the zone, which is defined as $SUM_z(t_1, t_2)$.

Given the Poisson arrival process predictions to set of vertices S_z of zone z, the value of $SUM_z(t_1, t_2)$, can be calculated as follows. As the pooled process of these independent Poisson processes is also a Poisson process and the intensity of the arrivals to zone z is $\lambda_z = \sum_{v \in S_z} \lambda_v$, where λ_v is arrival intensity to vertex v, the expected sum of waiting times at moment t_2 can therefore be estimated as $SUM_z(t_1, t_2) = \int_0^{(t_2-t_1)\lambda_z} (t_2 - t_1 - \frac{x}{\lambda_z})dx$. The index of the zone is defined as $I_z = T_z + SUM_z(t_1, t_2)$.

The last step of this method is to send the nearest available vehicle to the zone with maximal index.

4.7 Conclusion

In this chapter we developed functional architecture for electric autonomous taxi system management. The chapter discusses algorithmic solutions for different subproblems, designed in general problem setting 3.5. Remind, that the electric aTaxi management optimisation problem setting includes 3 main subproblems: charging management, empty vehicle redistribution and ride-sharing problems.

For the zone partitioning of the city network, the methods based on differential evolution were proposed. The problem setting was stated in order to make the obtained solutions applicable to taxi system optimisation, thus zones should be connected and have reasonable size.

In order to feed the optimisation methods with relevant heuristics, a set of techniques solving different subproblems occurring when optimising the SAV taxi system was investigated. The charging and parking stations were studied for the prediction of the possibility to find a place by autonomous taxis. Charging strategies for the vehicles based on the system states were proposed. New heuristical methods of empty vehicle redistribution algorithms for taxi services were investigated at both levels: at the vehicles level and at the zones level. Using a simple case, we evaluated three algorithms for four different cases of initial vehicle position and show that none of the algorithms is better, and that greedy types of algorithms often reach suboptimal outcomes. For the problem of the ride-sharing, a new zone-based algorithm for taxi services was investigated. The zone-based empty vehicle redistribution methods were investigated to provide more clear evidence of shifted traffic during rush hours. Such zone-based redistribution methods are necessary for such a shifted traffic, especially when using the ride-sharing.

Figure 4-7 summarises the relationships between the algorithmic solutions we have investigated. The arrow from method A to method B means, that method B uses method A. For example, one of necessary inputs for Spill Clustering Method (SCM) is Fill Clustering Method (FCM) output, and the FCM uses method of Differential Evolution, which is machine learning technique.

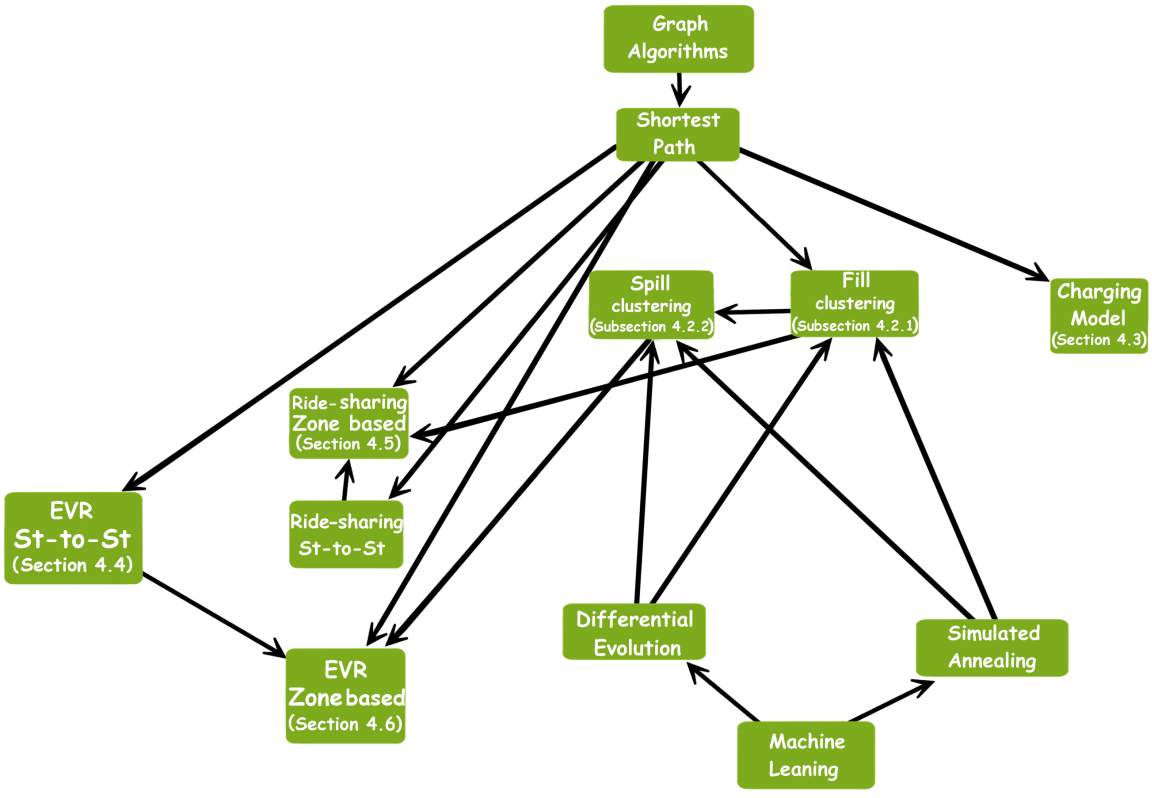


Figure 4-7 – Scheme of algorithmic solutions dependencies.

Chapter 5

Reinforcement learning for aTaxi system optimisation

5.1 Introduction

This chapter is devoted to the development of machine learning framework for the optimisation of the electric aTaxi system management (see Section 3.5). Reinforcement learning (RL) supports automation in the system where it is deployed. Unsupervised learning and deep learning can look similar to it. However, strategically these methods have differences, sometimes important ones, even if all these paradigms aim to replace humans and heuristics in different problems. So, why RL has been chosen as the main method in this work? The RL is very similar to the natural process of learning, in which the process or model acts and receives feedback about how it manages to cope with the problem: good or not.

In our context, the RL methods can be applied to various types of agents, such as vehicles or zones. Furthermore, large-scale agents, such as central system, are also possible. In connection with Chapter 4, we consider as one of inputs the zone decomposition, provided by FCM and SCM clustering methods (see Section 4.2).

This chapter is organised as follows.

In Section 5.2, in order to develop the framework of the reinforcement learning methods, we introduce the decision levels of vehicles we consider in the system and discuss different levels of scaling. We provide the main lines of the optimisation at these levels and reveal necessity of modules for solutions of the main subproblems highlighted in Section 3.5, such as charging management, empty vehicle redistribution, and dynamic ride-sharing problems.

Section 5.3 is dedicated to the vehicle-based and station-based reinforcement learning models. In the vehicle-based RL model, the memory of each vehicle, considered as an agent, is individual and vehicles do not share the information about the reward obtained under different conditions. In the station-based RL model, the learning information is shared by the vehicles in the location, and vehicles located in the same area have the same learning information. The second method is the basis of the microlevel, thus, the basis for the real-time small taxi system management.

In Section 5.4, zone-based reinforcement learning model is described. This method allows greater scalability under bigger granularity, thus, it is the basis for the real-time big taxi system management.

Section 5.5 provides the descriptions of the centralised reinforcement learning models, both station-based and zone-based. These methods can be used both in the real-life systems as well as in the prediction analysis for taxi systems.

Section 5.6 discusses the gap between Nash equilibrium and optimum, called the “price of anarchy”, and thus provides methods for aTaxi system management in the complete information case. These methods allow us to eliminate or to approach the global optimum in taxi systems in offline context. However, the requirement of the complete information, thus, the information about all future client arrivals and all their preferences, does not allow us to use this method in real time and only a-posteriori

knowledge is possible.

Finally, the conclusion in Section 5.7 summarises the dependencies between different methods discussed in this chapter, and provides areas of use and limitations of these methods.

5.2 Framework description and learning levels in the model

The formalisation of the system model described in Chapter 3 allows us to move forward for the modelling of the proposed solutions.

Generally, we can separate two main types of management in electric aTaxi system, notably centralised and decentralised ones. The decentralised management assumes in our model that the taxi system does not have an effect on the assignment of vehicles for clients. In such a management, the only role of the system is to provide the client demands to vehicles, and vehicles themselves can accept or reject them. In contrast, the centralisation of the management aims to control the full situation, thus, to assign relative vehicles to specific client demands.

Both methods have their own advantages and disadvantages. The decentralisation, on the one hand, can provide us more stable and robust system, which does not depend on the signal loss and the system failures. On the other hand, for demands made in advance, such a method can be less efficient, for example, if the vehicle accepts a call and then moves to another part of the city, planning to return later. The centralisation requires more specific real-time control techniques, and can at the same time provide more optimal management and become less robust because of computation time explosion.

In this thesis, we focus on non-stationary model-free management problem with distributed multi-agent approach [81]. For the solution, we use the approach of reinforcement learning. Generally, the multi-agent RL methods in transportation can be separated, for example, to cooperative and competitive ones [50, 144]. There is a difficulty, that a set of technical challenges appears in agent coordination [259, 144], both cooperative and competitive. Classical suitable methods to model-free systems are Linear Reward-Inaction and Linear Reward Penalty [261, 265, 8]. These learning algorithms are completely decentralised and can be thought of as a parallel distributed network similar to neural network models. In order to deal with occurring challenges, including ones related to joint actions, in Section 3.5 we propose to decompose the management problem with allocating classical subproblems for electric recharging management, ride-sharing, and empty vehicle redistribution. The algorithms of their solution are provided in Chapter 4. However, this is not the only way we propose to diminish the complexity of the management problem and to provide more stable solution.

In this chapter, we provide intermediate solutions at different granularity levels. The model of the system (Section 3.3) introduced the notation of zones in road network graph, and Section 4.2 provides the methods for zone decomposition for such a graph. These zones can replace the central system by separating the decision responsibility between them. In such a system, we deal with the optimisation in smaller systems which can help to deal with the curse of dimensionality.

We develop reinforcement learning methods at different decision levels, named microlevel, mesolevel, and macrolevel. Microlevel considers the smallest granularity, where the main decisions are to be provided by vehicles. Mesolevel is a step forward to bigger granularity and the scaling of the problem is achieved by the use of city partitioning in zones. Macrolevel considers the complete centralisation of the system and aims to provide the optimal solution for each time step, and is subdivided into different scaling levels. At the three levels, in order to deal with complex structure of the considered electric aTaxi system, we propose combined methods, aimed to decompose the problem. These combined methods are based on RL which manages client requests and assigns related vehicles to them. The RL method uses additional technics for electric recharging management, ride-sharing, and empty vehicle redistribution. The learning levels are summarised in Figure 5-1.

To obtain the price of anarchy [64] in the model, that is the difference between the global system optimum and the user equilibrium (or another obtained solution), we should find the best solution in the case where all the information about all the demands and user preferences is known in advance. There are 2 main methods to do so: in small networks, it is possible to find a brute force (exhaustive)

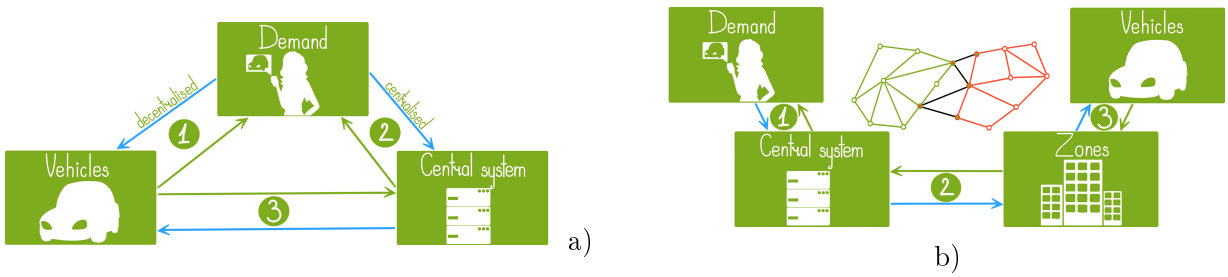


Figure 5-1 – a) The learning scheme for the decision level of vehicles: centralised and decentralised management. b) The learning scheme for the decision level of zones.

exact solution, whereas in medium and large networks, only heuristical solutions are possible.

Each of these 4 representations of the learning models is described in this chapter.

5.3 Microlevel: vehicle-based and station-based decision models

This section describes the reinforcement learning model at microlevel. We first describe the states of vehicles, the events, the actions of the vehicles, and the main lines of the learning algorithm we propose in Subsection 5.3.1. Subsection 5.3.2 provides the analysis of this method and its advantages and disadvantages. In Subsection 5.3.3 we propose an improved version of this vehicle-based reinforcement learning algorithm.

5.3.1 Vehicle behaviour in the system

We provide explicit analysis of the vehicle behaviour by studying the possible states of the vehicle, the possible events that can occur in the system and influence the vehicle, and finally the actions the vehicle can accomplish.

We consider the following general states of the vehicle:

- S1: *Idle* at a parking or charging station, without movement;
- S2: *Serving clients*. The more clients are in the vehicle, the bigger reward is;
- S3: *Charging* at the charging station;
- S4: *In empty ride* to meet the client;
- S5: *In empty ride* to a charging or a parking station;
- S6: *In empty ride* for heuristic reason such as meeting the predicted demand or displacement to another zone of the considered region;
- S7: *Fully discharged* in the street and not at a charging station.

However, every state defined above can have different substates, which depend on factors such as charging level, current occupancy etc. Nevertheless, we do not focus on the substates and consider the vehicle behaviour only in main lines.

In order to build the relevant reinforcement learning model, we provide all the possible connections and transitions between the different states described above. For that, we need to define the actions that may occur in the system. At any time, a vehicle can accomplish several possible strategies:

- E1: *Accept the call* occurring in the system.
- E2: *Decline the call.*
- E3: *Pick up the client at the station* if there is an available space in the vehicle.
- E4: *Land the client.* The client is served and sends its feedback on the finished trip.
- E5: *Get discharge warning.* The power level is low and the vehicle has to move to charging station. This warning appears based on chosen charging strategy, as described in Subsection 4.3.3.
- E6: *Get charged signal.* The vehicle is charged enough and ready to return to the system. This signal is based on chosen charging strategy (see Subsection 4.3.3).

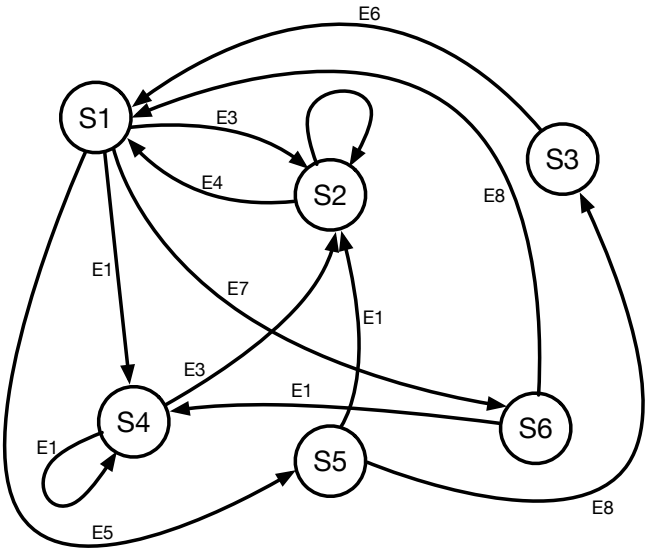


Figure 5-2 – Vehicle-based RL state machine.

- E7: *Make decision* in order to try to meet potential demand based on some heuristics. This option is possible only if enough experience had been already gained. For the potential demand serving, the methods described in Section 4.4 are used.
- E8: *Do nothing,* thus, to continue current action (current trip, charging or parking).

The states change with incoming actions can be represented as in Figure 5-2. Each arrow is accompanied by action indicating this transition. We do not draw arcs related to actions **E2** and **E8** which do not lead to state changing.

There are several cases where the vehicle has to make a decision:

1. Accept the call and possibly take a client, or reject the call?
2. Which level of left charge is sufficient to not miss client and to not get in the situation, where the sitting client cannot be delivered due to power shortage?
3. Which station to choose to move to when idle?

Each decision can penalise the vehicle if the client could not be delivered to its destination point. Moreover, the parameters of these decisions are subject to machine learning. Questions 2 and 3 are subject to heuristical methods because of the curse of dimensionality occurring when applying them to the reinforcement learning model. Therefore, we should take into account and use algorithmic methods from Chapter 4.

The complete information about the state-action space is described below.

- **S1: idle**.
  - Client call requires the vehicle to choose between **E1** and **E2**. The vehicle should calculate the objective function value for this call, that depends on current station, departure station, destination station and other parameters of the learning model, described in Subsection 5.3.2. The charging strategies are also included in the decision and if the vehicle is not able to finish the current trip sequence when adding a new client, and to move to the charging station after, the call will not be accepted. If vehicle chooses **E1**, the system or client will make the decision to accept or reject proposed ride. The accepted ride is represented as **E1** transition, and rejected one as **E2**. If the choice of accepting is right, the utility function for this ride is positive, and the vehicle gets a reward. Action **E1** leads to **S4**, whereas action **E2** does not change the state of vehicle.
  - Action **E3** fires, when the vehicle agrees to take the client at its current location.
  - Action **E5** fires, when the vehicle makes decision to move to charging station. As the cost of discharging during the trip is too high, this event is performed based on methods described

in Subsection 4.3.3. Therefore, this decision is to be made with probability 1 when the discharge warning is got by the system and the vehicle is idle. Other cases are checked at every decision point. For the same reason, state **S7** is not possible in the state machine.
  - Action **E7** fires, when idle vehicle makes a decision to go to another station, where, based on some heuristic or the some prediction, the demand is expected (EVR). Such a decision depends not only on current system state, but also on historical data, the predicted future demands and their parameters. This additional decision management is considered in Subsection 4.4. The elimination of action **E7** changes vehicle state to state **S6**.

- **S2: serving**.
  - Action **E1** may be fired if there is a free seat(s) in vehicle, ride-sharing is enabled, the client is located not far from the current trajectory, and the destination of the client is in acceptable area. The charging strategies are also included in the decision process. This action does not change the state of the vehicle.
  - Action **E4** can lead to state **S1** if the served client was the last client in the vehicle, otherwise the vehicle stays in state **S2**.

- **S3: charging**.
  - The only action that can change this state is **E6**. Action **E6** fires at one of the following conditions. "Demand condition" (see Subsection 4.3.3) means that there are more demands than available vehicles and the current charge level has reached critical value. "Strong condition" which means that when the vehicle is totally charged, the charging cycle should be terminated. When action **E6** fires, the state changes to **S1**.

- **S4: moving to client**.
  - Action **E1** may be fired if ride-sharing is enabled and all requirements are satisfied. The state changes to **S2** after picking up the client, thus, after firing action **E3**.

- **S5: moving to charge**.
  - Action **E1** may be fulfilled if the client is near current trajectory.
  - If there are no actions during the trip, the end of the trip leads to state change to **S3**.

- **S6: moving to position**.
  - Action **E1** leads to state **S4**.
  - If there are no actions during the trip, the end of the trip leads to state change to **S1**.

Well defined state-action machine makes it possible to base the learning using reinforcement learning methods as linear reward-inaction or linear reward-penalty. Indeed, every successful transition (such as **E4**) causes an increase in the corresponding coefficient in the environment (reward), and each unsuccessful one causes its decrease (penalty).

The above overview of states and events shows that an individual vehicle decision requires an environmental vector, which includes a sufficiently large set of coefficients. In the simplest case the required matrix size is $|V| \times |V|$, showing the acceptability of receiving a taxi request for a client who is in vertex $v_O \in V$ and is willing to reach vertex $v_D \in V$. For relevant results it requires the number of learning experiments noticeably larger than the size of this matrix — and this is for each vehicle. A shared set of environmental coefficients for all vehicles leads us to a different learning model, the centralised one (macrolevel).

Nevertheless, a decentralised learning mechanism is possible if the size of the environment coefficient space is reduced. For example, since this work is focused on electrical vehicles, the actual volume of the battery will vary as a function of the time. In this case, the environment space at each time can contain coefficients connecting the moving distance and the voltmeter reading. Moreover, each state transition could change these coefficients. Therefore, based on these considerations, as well as on the

fact that the price of detecting state **S7** for a vehicle is very high, the solutions based on the vehicle charge level are not taken into account in the learning model.

Taking all the discussed elements above into account, we provide a model-free online method for learning. Moreover, based on the states, and the possible rewards and penalties, we provide our own methodology for the learning process.

5.3.2 Vehicle-based learning algorithm

In this subsection we describe the learning strategy for the vehicle decisions. The probability of vehicle $\xi \in \Xi$ located at station $v_c \in V$ to accept the call from station $v_o \in V$ to station $v_d \in V$ is calculated as a linear function of the set of the seven following parameters (Table 5.1):

Table 5.1 – Parameters for the decision function

Parameter	Meaning
S_c^ξ	Correcting coefficient for client orders near station v_c.
A_o^ξ	Correcting coefficient for client departures from station v_o.
D_d^ξ	Correcting coefficient for client destinations station v_d.
W_o^ξ	Correcting coefficient for time window of clients in station v_o.
N_o^ξ	Number of demands from station v_o.
Np_o^ξ	Number of obtained rewards for client calls from station v_o.
B_o^ξ	Predicted time window size in station v_o.

Coefficient S_c^ξ takes into account the global constant C_{near}, which is the maximal allowed time to reach this station. Therefore, S_c^ξ is correcting coefficient for client orders in the circle of C_{near} with the centre in vertex v_c.

Thus, each vehicle saves the matrix of $|V| \times 7$ correcting coefficients, 7 for each vertex (station) $v_i \in V$ in the graph (system).

The value of predicted time window size B_o^ξ refers to its estimation for client requests from the station v_o. The correcting coefficient for time window W_o^ξ means sureness of time-window size value, when it equals to 1, it means that the real user time window will always match with the predicted time window size.

The correcting coefficients for departures (A_o^ξ) and destinations (D_d^ξ) refers to the attractiveness of this vertex as departure and destination station respectively. The bigger is this correcting coefficient, the more attractive this station is. This is related to the following considerations. If the demand from any station is large enough, then this station will be quite attractive as a destination station, since the probability of an empty run after arriving at this station will be low.

The number of requests N_o^ξ from station v_o is closely related to, in some opinion, its attractiveness and is also used in calculations.

In order to avoid the inaccuracies in predicting travel times associated with congestion, we introduce the correcting coefficient for vehicle ξ to calculate time-to-arrive T^ξ. This coefficient shows vehicle's estimation of the general congestion level in the system, because of the possibility of congestion propagation.

- Step 1. For each station $i \in V$, $S_i^\xi, D_i^\xi, A_i^\xi, W_i^\xi \leftarrow 1$, $N_i^\xi \leftarrow 0$, $B_i^\xi \leftarrow 600$. $T^\xi \leftarrow 1$.
- Step 2. Vehicle ξ is at station v_c.
- Step 3. The vehicle is idle or is moving and waiting for next request.
- Step 4. Request from station v_s is received for moving a client to station v_d (Figure 5-3).
- Let T_t be estimated trip time from v_c to v_s.
- Step 5. If the requirements to pass this step are not satisfied (see below), the reply is DECLINE.
- Step 6. If in previous step the answer was not DECLINE, let $P_{acc} = F(S_c^\xi, A_s^\xi, D_d^\xi, W_s^\xi, B_s^\xi, N_s^\xi, T_t \cdot T^\xi)$ be the vehicle probability to send an ACCEPT and to wait the decision of the client or the

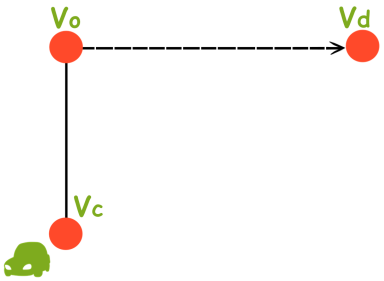


Figure 5-3 – Decision moment representation.

system. Note, that ACCEPT does not mean that the client of the request will be assigned to this vehicle.

- Step 7. If decision is DECLINE, then correcting coefficients $S_c^\xi, A_s^\xi, D_d^\xi, N_s^\xi$ are corrected as described below. Go to Step 3.
- Step 8. From here the decision is ACCEPT and the vehicle waits the decision of the system/client which is in state WAITING_FOR_ORDER (WFO).
- Step 9. If the reply from the client or the system is negative, then the client state for the vehicle switches to DECLINED and correction coefficients S_c^ξ, A_s^ξ and W_s^ξ are corrected with penalties.
- Step 10. If the reply from the client or the system is ACCEPT, then the client state for the vehicle switches to SERVING.
- Step 11. The vehicle schedules trip from v_c to v_s then from v_s to v_d and begins moving.
- Step 12. If the client leaves the system due to expired time window, W_s^ξ and B_s^ξ are corrected with penalties. The coefficient T^ξ is to be updated based on the predicted time on the way T_t, the current value of T^ξ and the real time on the way T_{real}. Go to Step 3.
- Step 13. If the client leaves the system because it is served by another vehicle, W_d^ξ is corrected with penalty, and coefficient T^ξ is updated based on obtained characteristics. Go to Step 3.
- Step 14. If the client was served but left a negative feedback, W_d^ξ and B_d^ξ are corrected with penalties. The relevant correcting coefficients are updated. Go to Step 3.
- Step 15. If the client was served and left a positive feedback, W_d^ξ and B_d^ξ are corrected with rewards. The relevant correcting coefficients are updated. Go to Step 3.

Conditions on which the reply is DECLINE in Step 5:

- charging level is not sufficient to perform the client's trip and then to move to nearest available charging station;
- there are no free seats when ride-sharing enabled request;
- the vehicle is already serving clients or wants to propose fully or partially shared trip, but somebody already in the vehicle or planned to be inside prohibits ride-sharing;
- the vehicle is charging;
- in case of already shared planned trip, the vehicle rejects all the clients which origin or destination are not near the way, as described in Section 4.2.2;
- the value $T_t > B_s^\xi$.

The correction operations for the steps above are described in the following.

- The correcting operations for Step 7:

Initial value	S_c^ξ	D_d^ξ	A_s^ξ	N_s^ξ
New value	$\frac{S_c^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{D_d^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{A_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$N_s^\xi + 1$

- The correcting operations for Step 9:

Initial value	S_c^ξ	W_s^ξ	A_s^ξ	N_s^ξ
New value	$\frac{S_c^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{W_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{A_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$N_s^\xi + 1$

- The correcting operations for Step 12:

Initial value	B_s^ξ	W_s^ξ	T^ξ	N_s^ξ
New value	$\frac{B_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{W_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$T^\xi + C_t \cdot \frac{T_{real} - T_t}{T_t}$	$N_s^\xi + 1$

- The correcting operations for Step 13:

Initial value	S_c^ξ	A_s^ξ	W_s^ξ	T^ξ	N_s^ξ
New value	$\frac{S_c^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{A_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{W_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$T^\xi + C_t \cdot \frac{T_{real} - T_t}{T_t}$	$N_s^\xi + 1$

- The correcting operations for Step 14:

Initial value	S_c^ξ	D_d^ξ	A_s^ξ	W_s^ξ	T^ξ	N_s^ξ
New value	$\frac{S_c^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{D_d^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{A_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$\frac{W_s^\xi \cdot N_s^\xi}{N_s^\xi + 1}$	$T^\xi + C_t \cdot \frac{T_{real} - T_t}{T_t}$	$N_s^\xi + 1$

- The correcting operations for Step 15:

Initial value	S_c^ξ	D_d^ξ	A_s^ξ	W_s^ξ	T^ξ	Np_s^ξ
New value	$\frac{S_c^\xi \cdot Np_s^\xi + C_s}{Np_s^\xi + 1}$	$\frac{D_d^\xi \cdot Np_s^\xi + C_d}{Np_s^\xi + 1}$	$\frac{A_s^\xi \cdot Np_s^\xi + C_a}{Np_s^\xi + 1}$	$\frac{W_s^\xi \cdot Np_s^\xi + C_w}{Np_s^\xi + 1}$	$T^\xi + C_t \cdot \frac{T_{real} - T_t}{T_t}$	$Np_s^\xi + 1$

- Function F, used in Step 6, is as follows:

$$F(S_c^\xi, A_s^\xi, D_d^\xi, W_s^\xi, B_s^\xi, N_s^\xi, S_d^\xi, t^\xi) = \frac{S_c^\xi \cdot C_{FS} + A_s^\xi \cdot C_{FA} + D_d^\xi \cdot C_{FD} + S_d^\xi \cdot C_{FOD} + W_d^\xi \cdot C_{FW}}{C_{FS} + C_{FA} + C_{FD} + C_{FOD} + C_{FW}}. \quad (5.3.1)$$

Coefficients $C_t, C_s, C_d, C_a, C_w, C_{CS}, C_{FD}, C_{FOD}, C_{FA}, C_{FW}$, hereinafter referred to as $\{C^s\}$ are global for the algorithm and calculated in the evaluation by differential evolution machine learning methods on synthetic data. One of these methods has the known exact solution (see Sections 6.5.2 and 6.5.1). These coefficients are weights for learning events. If coefficient C_1 is greater than coefficient C_2, the influence of coefficient C_1 on the resulting value of the function is greater, than the influence of C_2.

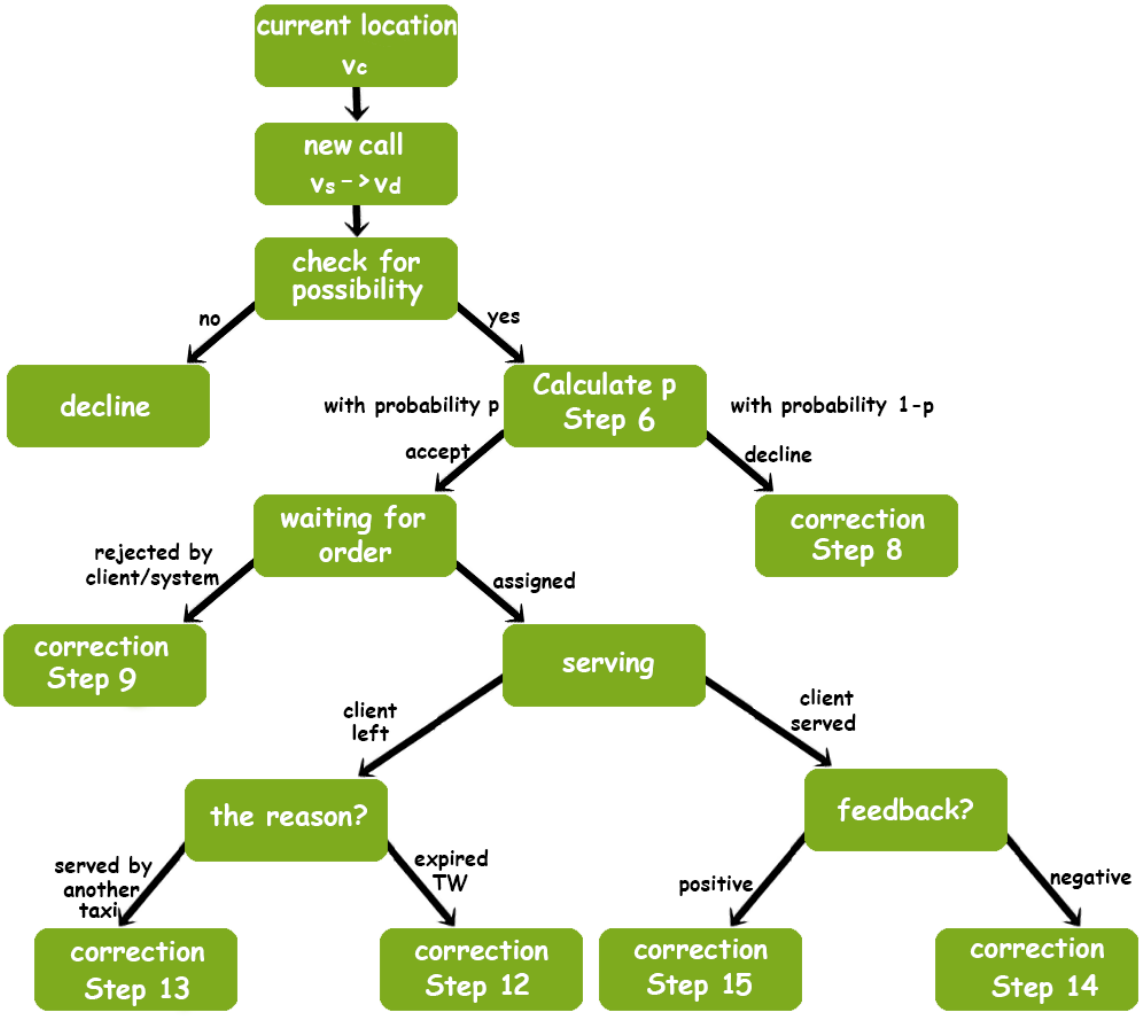


Figure 5-4 – The scheme of the vehicle-based learning algorithm.

The scheme of the vehicle-based learning algorithm is represented in Figure 5-4.

Advantages and disadvantages of vehicle-based RL model

Potential advantages of the vehicle-based RL model are the following:

- It is the easiest way to represent the city using the reinforcement learning, where agents are vehicles. The model can be easily evaluated by other algorithms implementation.
- There are no joint-actions, thus, the vehicles act independently. The action and state dimensions are smaller than in cases with shared learning information.

The potential disadvantages of the vehicle-based RL model are the following:

- The model takes a lot of steps to learn. There are no strong guarantee that such a model will adapt fast enough to the changes in the road situation or the client demand.
- The model can be expensive memory-wise and have big computational complexity.

5.3.3 Station-based reinforcement learning model

In order to avoid the disadvantages of the vehicle-based model, we can take into account, that different vehicles in the same initial conditions should take identical decisions. Therefore, the final version of the microlevel model is changed to the station-based model.

The correcting coefficients S_i, A_i, D_i, W_i, N_i, and B_i are shared between all the vehicles that are in the same positions. Thus, this learning method can be also considered as station-based, changing

the vehicle coefficients to station ones. The coefficients $C_t, C_s, C_d, C_a, C_w, C_{CS}, C_{FD}, C_{FOD}, C_{FA}, C_{FW}$ (Table 5.1) are recalculated based on differential evolution method on the artificial data. In order to avoid multiple assignments for one station, the original method is changed. The correcting operations are performed in the following cases:

1. If the ride was proposed to the client, but the client or the system has not chosen this ride, the correcting operations are as in Step 10.
2. If the ride was accepted by the client, but this one has left the system due to expired time window, the correcting operations are as in Step 13.
3. If the ride was accepted by the client, but the client has left the system because served by another vehicle, the correcting operations are as in Step 14.
4. If the client was served but left a negative feedback, the correcting operations are as in Step 15.
5. If the client was served but left a positive feedback, the correcting operations are as in Step 16.
6. If the ride was rejected by the system, the correcting operations are as in Step 8.

The arrival time of the vehicle to the client location is taken into account only at the moment of the decision making.

In such a model, the computational complexity remains the same, but the memory consumption is lower and the convergence characteristics outperform the ones of the vehicle-based model (see Chapter 7), and the more vehicles are in the system, the more significant is the difference. As a limit case, for small networks it is proportional to the amount of vehicles.

Hereinafter, we distinguish the vehicle-based RL model and the station-based RL model in the framework of the microlevel RL model.

5.3.4 Convergence

The important characteristics of any learning model are its convergence as well as its convergence time. There is a number of different forms of convergence that have been examined in the literature. For example, game theory approaches include roughly increasing in strength, average reward, empirical distribution of actions, expected reward, and strategies. However, in the reinforcement learning related to the vehicle movement optimisation, there is no clear criterion derived to evaluate the convergence of the learning process, and only for specific learning methods such as Q-learning, the definition of the convergence is defined [245].

In the considered multiagent approach, each agent learns either independently or using central system data. The learning results obtained by one agent can change the learning environment for other agents. In this work we consider the convergence in terms of the number of satisfied clients. However, the stochastic nature of the client utilities as well as combined algorithms randomisation, for example, for empty vehicle redistribution, do not allow us to reach with certainty the constant value. Nevertheless, even results in terms of classical oscillation theory cannot be reachable.

Let us provide the following definitions of **noised oscillation** and **noised convergence**.

Definition 5.1. *The process F is considered to be **noisy converged** in terms of the value $F(t), t \geqslant 0$, if the following condition holds:*

$$\exists T_0 \geqslant 0 : \exists C_0, C_1 \in \mathbf{R} : \forall t \geqslant T_0 \quad C_0 \leqslant F(t) \leqslant C_1.$$

It follows, the definition of noised convergence is closely related to the definition of bounded function. More precisely, the function is noisy converged if and only if it is bounded in some way $[T_0, \infty)$.

Definition 5.2. *The **amplitude** of the noisy converging process, in terms of its value $F(t), t \geqslant 0$, is as follows:*

$$\min_{C_0^*, C_1^*} (C_1^* - C_0^*) : \exists T_0 \geqslant 0 : \forall t \geqslant T_0 \quad C_0^* \leqslant F(t) \leqslant C_1^*, \quad C_0^*, C_1^* \in \mathbf{R}.$$

We call C_0^* the **lower bound** of the oscillation and C_1^* its **upper bound**.

Definition 5.3. *The **converged learning result** of the noisy converging process F is a mean value of its lower and upper oscillation bounds.*

Definition 5.4. *The **convergence time** of the noisy converging process F with lower bound C_0^* and upper bound C_1^* is defined as follows:*

$$\min_{T_0 \geqslant 0}(T_0) : \forall t \geqslant T_0 \quad C_0^* \leqslant F(t) \leqslant C_1^*.$$

In a discrete process F, we refer to the convergence time as to number of iterations to enter the converged state ItC.

Hereinafter, if the opposite is not specified, we refer to noised oscillation and noised convergence as oscillation and convergence, respectively.

Summary

The microlevel, which is the simplest and most classical method of representing the elements of a transport system in optimisation problems, is analysed in this section from two points of view, depending on learning information. The first method, vehicle-based one, is a method of individualised learning, more similar to the learning of human taxi drivers, and the absence of shared information is meant. The second method, station-based one, is the method of shared learning, that is, vehicles located in the same place (station) have the same learning information. In the framework of this thesis, we consider the first method only for the comparison with the second one. The second method is chosen as the basis of the microlevel and further, unless otherwise specified, by microlevel we mean the station-based reinforcement learning.

5.4 Mesolevel: zone-based decision model

This section describes zone-based reinforcement learning model. Here, the agents are the zones in the system, and vehicles are the moving resources associated with them. The microlevel learning model requires huge learning information, therefore when applied to big networks, the convergence time of the learning can be non-adequately big [144]. In addition, when the information changes, for example because of the disruption, the vehicle-based or station-based learning model are not able to react to it fast enough for the system functioning in real time.

To lessen the computation complexity and learning characteristics of the initial model and to make the learning model more scalable, the zone-based learning model is proposed. The zones in this model are obtained by Spill clustering method, discussed in Subsection 4.2.2, that provides connected uniform overlapped clusters, called hereinafter "zones".

At this level, the main objective to learn is to determine and provide the probability to accept the call, based on the origin zone, the destination zone, the number and characteristics of available vehicles, and the gained experience. For this purpose, it is necessary to develop appropriate relevant methods for processing client requests, as well as methods of fleet management.

The general methodology for processing requests from clients is as follows. The demand arriving to the system is sent to the relevant zones of the system and the zones themselves make an acceptance or rejection decision of the call. The zone receiving a demand makes a pre-call to all the vehicles inside it and obtains the list of vehicles accepting this pre-call with their proposed ride information. Thus, the acceptance or rejection of the call by a zone is accompanied by potential parameters in case of acceptance, as the number of vehicles that can propose a shared ride and the total number of vehicles. If several zones accept the call, the more suitable zone will be chosen by the system. It follows that the zone, as an agent, will choose the more suitable vehicle to send to the client.

If a zone, based on its logic, receives a request from a client, but does not have enough vehicles to serve this request, eventually it can make a call to adjacent zones so that adjacent zones send an available vehicle to it. The probability that it will get the vehicle suitable to client request time can

be determined from the learning information. Thus, zones can store information for other zones, such as statistics on accepted and refused requests for empty vehicles. However, we assume that in this case, this request should have initially been served by a zone that could send an available vehicle to the original zone.

Summarising the above, the second core difference between the microlevel models and the zone-based model, in addition to computational complexity, is the number of the events occurring in the system which is different for each level. Moreover, the zone-based model can ignore undesirable events that are locally optimal for vehicles (microlevel). The aim is to increase the number of good events (successfully served clients) and to reduce the number of bad ones (non served clients or served out of their utility constraints).

Let us consider in more details what exactly can zones store and process at the mesolevel.

5.4.1 Zone characteristics and actions

In order to avoid the increasing complexity of the learning model, the movement of empty vehicles between zones is carried out by heuristic algorithmic methods described in Section 4.6. Thus, the EVR is carried out with a predetermined regularity, choosing between zone index-based EVR and surplus/deficit zone-based EVR methods (see Subsections 4.6.1 and 4.6.2). Both of these methods provide us a calculation output the origin and destination zones. The proposed ride is performed by the system as a new demand with corresponding parameters. In this case, the destination station indicates the station which, on average, is the closest in time to be reached from the origin zone. If there are several such stations, then the station is randomly uniformly selected among them.

Zone actions at mesolevel are binary and occur only when there is a request from a client. If there is a request in the system, the actions to provide are ACCEPT or REJECT, having previously checked vehicles availability in this zone. In the case of acceptance, the zone ranks available vehicles, based on the following considerations. The rank is based on the smallest empty mileage with the limitation of the possibility to serve the request based on the battery level and the client's constraints. When the zone is chosen to serve the request, the zone provides the vehicle based on its rank. Shared trips are prioritised in the model, thus, if there are vehicles that can theoretically serve the trip with ride-sharing within client constraints, the vehicle will be chosen within this set.

5.4.2 Central system level

In this model the central system acts based on the following algorithmic strategies.

The client calls arriving to the system are sent by the central system to relevant zones within a certain radius from the origin of the required trip. Each concerned zone $z \in Z$ evaluates all the vehicles inside it and builds a list of available vehicles in this zone $Avail(z, p)$ to serve client $p \in P$, based on their occupancy and battery level. This set includes subset $Availsh(z, p)$ of vehicles which can provide shared trip for this client. The zones, as agents, learn to take a decision, which consists of the acceptance or rejection of the client call. This decision is provided as calculated probability to accept the call, given the origin and destination zones, and set $Avail(z, p)$, based on gained experience.

After receiving the decision of each concerned zone, the central system chooses, from accepting zones, the zone with minimal distance between the available vehicle and the client location. If there are few such zones, the system chooses the one with the biggest value of $k \cdot Availsh(z, p) + Avail(z, p)$, where $k \geqslant 0$ is a coefficient of the ride-sharing priority. If no such a zone is found, the central system declines the client call. In addition, the central system also provides the information for zone-based EVR and for ride-sharing methods, as described in Section 4.2.2. The EVR is carried out with a predetermined regularity, choosing between methods for zone-based IBR and surplus/deficit zone-based EVR. Both of these methods provide us the origin and destination zones for the redistribution.

5.4.3 Learning algorithm

The learning strategy for zones can be described as follows. The probability of zone z_c to accept the call from zone z_o to zone z_d is calculated as a linear function of the set of the seven following parameters:

- S_c^z — correcting coefficient for client orders near zone z_c.
- A_o^z — correcting coefficient for client departure zone z_o.
- D_d^z — correcting coefficient for client destination zone z_d.
- W_o^z — correcting coefficient for time window of clients in zone z_o.
- N_o^z — number of demands from zone z_o.
- Np_o^z — number of successfully served demands from zone z_o.
- B_o^z — predicted time window size in zone z_o. The initial value is equal to 10 minutes based on literature [122].

These characteristics are the same as for the microlevel model (see Table 5.1) except that instead of the vehicle and the station correcting coefficients we have zone correcting coefficients. Moreover, the station-based RL method can be considered as a limit transition of a zone-based method, when the size of each zone is supposed to be equal to one station.

Thus, each zone saves the matrix of $|Z| \times 7$ correcting coefficients, 7 for each zone in the system. We define the correcting coefficient of zone z to calculate time-to-arrive as T^z, and it signifies the zone opinion of the congestion level in the system.

The learning algorithm for zone z can be thus described as follows.

- Step 1. For each zone $z_i \in Z$, $S_i^z, D_i^z, A_i^z, W_i^z \leftarrow 1$, $N_i^z \leftarrow 0$, $B_i^z \leftarrow 600$. $T^z \leftarrow 1$.
- Step 2. Zone z_c is waiting for the next request.
- Step 3. Request p from zone z_s is received for moving the client to zone z_d.
- Let T_t be the estimated trip time from zone z_c to the client in zone z_s. This is the average travel time between each station of zone z_c to the current client's station.
- Step 4. The zone calculates the numbers $Avail(z_c, p)$ and $Availsh(z_c, p)$ for the demand.
- Step 5. If $Avail(z_c, p) = 0$, the demand can not be accepted and the zone reply is DECLINE.
- Step 6. Let $P_{acc} = \min\left(1, G(S_c^z, A_s^z, D_d^z, W_s^z, B_s^z, N_s^z, T_t \cdot T^z, Avail(z_c, p), Availsh(z_c, p))\right)$, where G is defined below. With probability P_{acc}, zone sends an ACCEPT and provides the numbers $Avail(z, p)$ and $Availsh(z, p)$ for the central system. Note, that ACCEPT does not mean that the client will be assigned to this zone by the central system.
- Step 7. If decision is DECLINE, then coefficients $S_c^z, A_s^z, D_d^z, N_s^z$ are corrected as described below. Go to Step 2.
- Step 8. From this step, the decision is ACCEPT and zone assigns to the central system state WAITING_FOR_ORDER (WFO), meaning waiting for the decision of the central system.
- Step 9. If the system reply for zone ACCEPT is negative for zone in WFO state, then central system state for the zone is to be switched to DECLINED, and S_c^a, A_s^a and W_s^a are corrected with penalties.
- Step 10. If the system reply for zone ACCEPT is positive and client is in WFO state, then its state for the zone switched to TOSERVE.
- Step 11. The zone searches the vehicle to serve the client based on the heuristics described above. On this step only the vehicles capable to perform the request based on their charging level and clients restrictions are selected. From these vehicles the vehicle with minimal empty run is preferred. Shared trips are prioritised in the model, thus, if there are several vehicles that can theoretically serve the trip with providing of the ride-sharing within client constraints, including maximal delay, the vehicle will be chosen within this set. Sharing of rides is provided based on methods described in Subsection 4.5. It is however just a candidate ride and the client can accept or decline it. In case of declined ride all coefficients get penalties and zone switches to Step 2.
- Step 12. The zone assigns selected vehicle adding a new trip with noted changes.
- Step 13. If the client leaves the system due to expired time window before the assigned vehicle reaches it, W_s^z and B_s^z are corrected with penalties. The coefficient T^z has to be updated based

on predicted time in the way T_t, current value of T^z and real time in the way T_{real}. Go to Step 2.

- Step 14. If the client leaves the system because served by another vehicle, W_d^z is corrected with penalty, and coefficient T^z is updated based on obtained characteristics. The zone ratings are penalized. Go to Step 2.
- Step 15. If the client was served but left a negative feedback, W_d^z and B_d^z are corrected with penalties. The zone correcting coefficients are updated. Go to Step 2.
- Step 16. If the client was served and left a positive feedback, W_d^z and B_d^z are corrected with rewards. The zone correcting coefficients are updated. Go to Step 2.
- The correcting operations for Step 7 are the following:

Initial value	S_c^z	D_d^z	A_s^z	N_s^z
New value	$\frac{S_c^z \cdot N_s^z}{N_s^z+1}$	$\frac{D_d^z \cdot N_s^z}{N_s^z+1}$	$\frac{A_s^z \cdot N_s^z}{N_s^z+1}$	N_s^z+1

- The correcting operations for Step 9 are the following:

Initial value	S_c^z	W_s^z	A_s^z	N_s^z
New value	$\frac{S_c^z \cdot N_s^z}{N_s^z+1}$	$\frac{W_s^z \cdot N_s^z}{N_s^z+1}$	$\frac{A_s^z \cdot N_s^z}{N_s^z+1}$	N_s^z+1

- The correcting operations for Step 11 are the following:

Initial value	S_c^z	A_s^z	W_s^z	B_s^z	N_s^z
New value	$\frac{S_c^z \cdot N_s^z}{N_s^z+1}$	$\frac{A_s^z \cdot N_s^z}{N_s^z+1}$	$\frac{W_s^z \cdot N_s^z}{N_s^z+1}$	$\frac{B_s^z \cdot N_s^z}{N_s^z+1}$	N_s^z+1

- The correcting operations for Step 13 are the following:

Initial value	B_s^z	W_s^z	T^z	N_s^z
New value	$\frac{B_s^z \cdot N_s^z}{N_s^z+1}$	$\frac{W_s^z \cdot N_s^z}{N_s^z+1}$	$T^z + C_t^* \cdot \frac{T_{real}-T_t}{T_t}$	N_s^z+1

- The correcting operations for Step 14 are the following:

Initial value	S_c^z	A_s^z	W_s^z	T^z	N_s^z
New value	$\frac{S_c^z \cdot N_s^z}{N_s^z+1}$	$\frac{A_s^z \cdot N_s^z}{N_s^z+1}$	$\frac{W_s^z \cdot N_s^z}{N_s^z+1}$	$T^z + C_t^* \cdot \frac{T_{real}-T_t}{T_t}$	N_s^z+1

- The correcting operations for Step 15 are the following:

Initial value	S_c^z	D_d^z	A_s^z	W_s^z	T^z	N_s^z
New value	$\frac{S_c^z \cdot N_s^z}{N_s^z+1}$	$\frac{D_d^z \cdot N_s^z}{N_s^z+1}$	$\frac{A_s^z \cdot N_s^z}{N_s^z+1}$	$\frac{W_s^z \cdot N_s^z}{N_s^z+1}$	$T^z + C_t^* \cdot \frac{T_{real}-T_t}{T_t}$	N_s^z+1

- The correcting operations for Step 16 are the following:

Initial value	S_c^z	D_d^z	A_s^z	W_s^z	T^z	Np_s^z
New value	$\frac{S_c^z \cdot Np_s^z + C_s^*}{Np_s^z+1}$	$\frac{D_d^z \cdot Np_s^z + C_d^*}{Np_s^z+1}$	$\frac{A_s^z \cdot Np_s^z + C_a^*}{Np_s^z+1}$	$\frac{W_s^z \cdot Np_s^z + C_w^*}{Np_s^z+1}$	$T^z + C_t^* \cdot \frac{T_{real}-T_t}{T_t}$	Np_s^z+1

Function G, used in Step 6 is as follows:

$$G(S_c^z, A_s^z, D_d^z, W_s^z, B_s^z, N_s^z, t^z, Avail(z,p), Availsh(z,p)) = \tag{5.4.1}$$
$$\frac{S_c^z \cdot C_{FS}^* + A_s^z \cdot C_{FA}^* + D_d^z \cdot C_{FD}^* + S_d^z \cdot C_{FOD}^* + W_d^z \cdot C_{FW}^* + \frac{k \cdot Availsh(z,p) + Avail(z,p)}{Avail(z,p)} \cdot C_{AV}}{C_{FS}^* + C_{FA}^* + C_{FD}^* + C_{FOD}^* + C_{FW}^* + C_{AV}}.$$

Coefficients $C_t^*, C_s^*, C_d^*, C_a^*, C_w^*, C_{CS}^*, C_{FD}^*, C_{FA}^*, C_{FOD}^*, C_{FW}^*$ are global for algorithm and in this method are supposed to be equal to the corresponding coefficients $C_t, C_s, C_d, C_a, C_w, C_{CS}, C_{FD}, C_{FA}, C_{FOD}, C_{FW}$ in the station-based reinforcement learning (see Section 5.3). The available vehicles influence the function based on their sharing opportunities. Thus, the coefficients k and C_{AV}, which show influence of the available vehicles in the zone as well as the ride-sharing preferences of the system, are new and unique for this model and are calculated based on differential evolution machine learning method for the training data set of the city of Porto, which is the biggest of evaluated networks, discussed in Subsection 6.5.5. Hereinafter we refer to the considered coefficients set as to $\{C^z\}$.

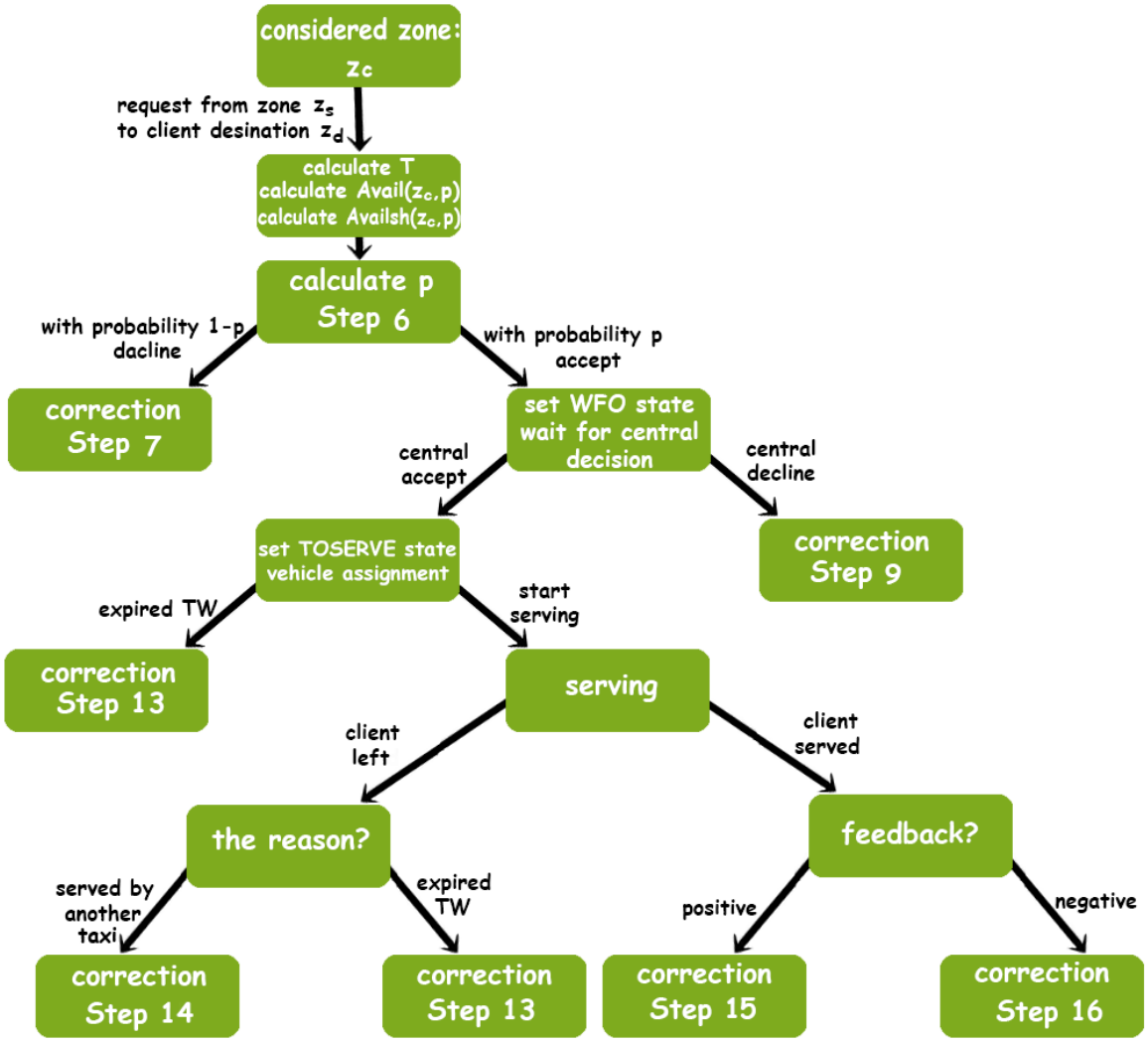


Figure 5-5 – The scheme of the zone-based learning algorithm.

The scheme of the zone-based learning algorithm is represented in Figure 5-5.

Summary

The proposed learning model provides the transition to greater scalability, which is especially important when modelling systems of large dimensions. This solution is designed for use in transport networks of real-life size and characteristics. Of course, the implementation of this model requires massive preparatory work, including the clustering of the transport graph, as described in Section 4.2. The method described above uses heuristics of charging, ride-sharing and empty vehicle redistribution. It is the main contribution of this work, since it can be both provided in real-time simulation and optimisation, as well as because of its learning characteristics.

Advantages and disadvantages of mesolevel method.

Potential advantages of the method compared to the microlevel methods are the following:

- **Scalability:** the model can be scaled using different zone decomposition.
- **Resources economy:** it requires less resources than vehicle agent-based model.
- **Learnability:** it prones to learn faster than models with smaller granularity.
- **Extensibility:** it allows adding external optimisation modules, based on graph structure, in order to approach the system optimum.

The potential disadvantages of the method are the following:

- **Granularity:** the big granularity of the road network graph can lead to less exact strategies especially in repeating pattern-based situations.
- **Complexity:** the realisation of this model is more complex, as it requires event-based modelling.

Disregarding the big granularity, it can influence insignificantly the optimal results. The realisation of the event-based model requires more time and effort. However, it can allow faster simulation with better quality, including the real-time simulation. In the end, it can lead to the economy of human and computing resources. The results in Section 7.7 show the improvement in execution time, up to 5 times, without loss in quality of service.

5.5 Macrolevel: centralised decision models

In this section, we discuss the methods of the centralised taxi management. These methods aim to reach the local system optimum at every time step.

5.5.1 Centralised Station-based model

Challenges linked to centrality

The sharing of all the information between all the vehicles in centralised system can cause the curse of dimensionality. Indeed, in the vehicle-based reinforcement learning model discussed in Subsection 5.3.2, each agent a, that is vehicle or station, saves for each station v_i seven parameters (S_i^a, A_i^a, D_i^a, W_i^a, N_i^a, Np_i^a, and B_i^a). When we share all the information and take them all into account, different challenges [176] appear in such a multi-agent reinforcement learning. These challenges in terms of the considered aTaxi system are the following:

- Problem setting feasibility: In the vehicle-based RL model, the sequence of actions from the chosen policy is based on the reward and penalty signals from the environment. If we model a fully centralised system, the action space can be significantly larger than the one of vehicle-based or station-based model. For example, during the empty vehicle redistribution, the system must determine several parameters, including the number of vehicle available to redistribute. Taking the ride-sharing into account, the system needs to solve the multi-variable optimisation problem at each optimisation step, and to take into account the learning information, position, and planned rides of each vehicle in the system.

- Large-scale agents: training a large number of agents using RL methods is again challenging. The environment for each agent is non-stationary since other agents are learning and affecting the environment at the same time. Thus, the best action at one moment can lead to the penalties at other moments because of the lack of information.

- Coordinations and context dependence of action space: the action space is changing dynamically because the agents (vehicles or stations) are acting subject to different locations and stations in the city. Thus, the number of feasible actions depends on the geographic context of the location. For example, at the charging stations, only small set of actions is available. The coordination of agents is another challenge. Since the optimisation problem should be solved, a question arises: is it preferable to take more clients on shared trip, even if there is a price to pay, for example, one additional minute of client waiting?

Shared information at central level

Taking these facts into account, and the curse of dimensionality occurring in the original problem setting, in the centralised case we use the modification of the station-based reinforcement learning. As in previous sections, the empty vehicle redistribution, ride-sharing, and charging are based on heuristical methods described in Chapter 4.

In the centralised model, the information about the congestion level is fully known by the central system and this parameter is not of interest to learn in this model.

The learning parameters, which are fully shared, are the following:

- S_i — correcting coefficient for client orders near station v_i, $S_i \geqslant 0$.
- A_i — correcting coefficient for client departure station v_i, $A_i \geqslant 0$.
- D_i — correcting coefficient for client destination station v_i, $D_i \geqslant 0$.
- W_i — correcting coefficient for time window of clients in station v_i, $W_i \geqslant 0$.
- N_i — number of requests from station v_i, $N_i \in \mathbb{N}_0$.
- Np_i — number of positive results from station v_i, $N_i \in \mathbb{N}_0$.
- B_i — predicted time window size in station v_i, $B_i \geqslant 0$.

The learning strategy therefore is the same as the one described in Subsection 5.3.3. The only difference in this strategy is the shareability of the parameters. However, the decision process at the central system level differs in the following way.

The central system sends the demand to all the vehicles in the area of origin of the demand. The initial position of the vehicle is taken into account in the learning function as well as in the arrival time prediction. Then, every vehicle takes a decision based only on heuristics, thus, guided only by considerations of the possibility of providing this ride. Finally getting a set of available vehicles, the central system solves a local optimisation subproblem, minimising the empty rides based on the matching strategies, and prioritising the shared trips if the client accepts ride-sharing.

Rematching problem

The centralised nature of the system level can provide the following improvement in order to minimize the energy consumption: switch the scheduled rides, when the vehicles are empty. If this switch diminishes the empty ride mileage, then the system provides such switchings (re-scheduling).

These re-scheduling are provided based on the bipartite graph methods. Here, the bipartite graph is constructed the following way. The 2 families of vertices are respectively idle vehicle locations $\{\xi\}$ (here, we consider an empty vehicle moving to a client to be idle) and the scheduled first destinations $\{d\}$. Weights in such a graph are relative energy consumptions $\{\xi_i, d_j\}$ for vehicle ξ_i to reach destination d_j. The additional constraints based on the battery levels and clients time windows are also provided in the model as the lack of the relative edges, as shown in Figure 5-6. For the possibility of using the Hungarian algorithm [159], the missing edges can be considered to be edges of infinity (huge) weight. The objective of such a subproblem is to provide a set of edges such that every vertex in set $\{d\}$ belongs to exactly one chosen edge, each vertex in set $\{\xi\}$ belongs to not more than one chosen edge, and the

sum of weights of chosen edges is minimal. The problem setting (setting of first destinations) ensures that $|\{d\}| \leqslant |\{\xi\}|$.

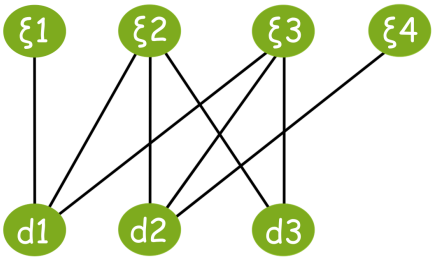


Figure 5-6 – The bipartite graph for the rematching reallocation.

The values of $|\{\xi\}|$ and $|\{d\}|$ can differ significantly, thus, the RJVI algorithm, the improved in [38] rectangular variant of LAPJV, is chosen for the solution of re-matching problem.

5.5.2 Centralised Zone-based model

Zone-based centralised model, as well as all the models from the class of centralised ones, consider the case of complete information about system state at each moment (but not about future demands).

In this model, central system takes into account all the current information including current demand in the system and vehicle characteristics. The congestion level in the system is considered to be known by the system, and zones do not learn to predict it.

The main advantages of a centralised model are the following. The empty vehicle redistribution can be provided in more exact way, taking into account the overall information. The zone-based ride-sharing can also be provided more exactly, taking into account the current demand from other zones, without excessive exchanges between zones. Thereby, the central system can manage the vehicle movement, knowing the general system performance.

The learning parameters, which are fully shared, are the following:

- S_i — correcting coefficient for client orders near zone z_i, $S_i \geqslant 0$.
- A_i — correcting coefficient for client departure zone z_i, $A_i \geqslant 0$.
- D_i — correcting coefficient for client destinations zone z_i, $D_i \geqslant 0$.
- W_i — correcting coefficient for time window of clients in zone z_i, $W_i \geqslant 0$.
- N_i — number of demands from zone z_i, $N_i \in \mathbb{N}_0$.
- Np_i — number of positive results from zone z_i, $N_i \in \mathbb{N}_0$.
- B_i — predicted time window size in zone z_i, $B_i \geqslant 0$.

The learning algorithm therefore is the same as the one described in Subsection 5.4.3. The only difference is the shareability of the parameters. However, the decision process at the central system level differs in the following way. The central system accepts demands and sends them to zones. Then it selects a zone for service. Zones make a service decision based on fully centralised parameters, and the correcting coefficients of origin and destination, as described in Section 5.4. The selection of the zone to serve the demand takes into account the overall demand and, if necessary, the more perspective demand has to be prioritised. When the central system selects a service zone, the local optimisation problem should be solved. If there are several requests in the system, the problem of multi-match on a bipartite graph is solved, which is constructed as follows.

The needed bipartite graph contains two partitions. First partition includes only zones where there are available vehicles from accepted demand zones that can serve the demands $\{\xi\}$. Second partition includes only vertices of base graph where there are client demands $\{d\}$. The edges in this graph are constructed from vertices belonging to first partition to vertices belonging to second partition. The weight of edges are equal to the travel times from the zone to the demand. As in Subsection 5.5.1, if zone rejects a demand, the weights are considered to be infinite. If any vehicle from the available zone is not able to serve the demand due to battery/ride-sharing constraints, the relative edge weight

is also considered to be equal to infinity. The assignment problem has to be solved as described above. Then the system chooses for each demand the most suitable zone. Such a double-setting, with zone acceptance of the demands based on its learning model, and the central system that optimises the total performance based on vehicles, allows obtaining more precise solutions.

A zone-based EVR function is invoked regularly with an intensity depending on the number of demands in the system and the number of available vehicles,

5.6 Complete information and the price of anarchy

The offline context of the electric aTaxi system management problem setting assumes that all the client demands arrive at the start of the simulation. The optimisation problem, which takes into account all the factors such as charging, ride-sharing, empty vehicle redistribution, belongs to the class of NP-hard problems (see Chapter 2). In our knowledge (see Subsection 2.3.3), there are no clear heuristical methods to reach the global optimum in the electric shared taxi systems in the case of complete information, and only heuristics that aim to achieve the social optimum in particular moments are used. Therefore, in this section we propose a new technique that aims to move forward global optimum, but not necessary to reach it.

5.6.1 Exact method of system optimum search

We propose the following method of the global optimum search. This method is based on the search tree pruning method [209]. The search tree is constructed as follows. The vertices in this tree are the decision moments and the edges are the relevant possible actions.

For the decision moment of a new client arrival, thus, of a new demand, the subtree is as follows:

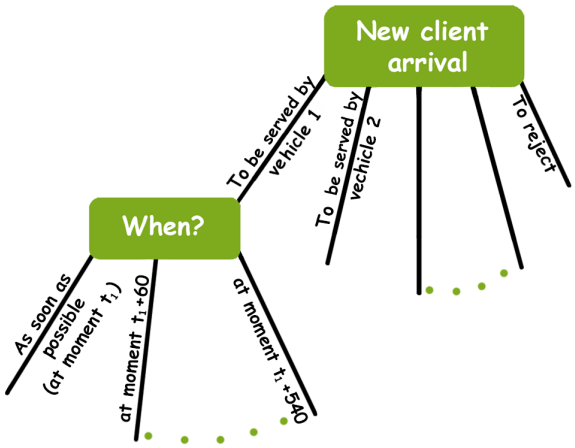


Figure 5-7 – Decision subtree at client arrival moment.

Some of the considered decisions can be impossible because of the constraints related to the time window of the client, the ride-sharing parameters, or the charging. In Figure 5-7 the client with the time window of 10 minutes is considered. In this example, the discretisation of 1 minute for the pick-up time leads to 10 possible edges.

The charging and parking strategies in this tree are realised as additional vertices occurring regularly for each vehicle (where frequency depends on the battery parameters). The corresponding subtree is as follows:

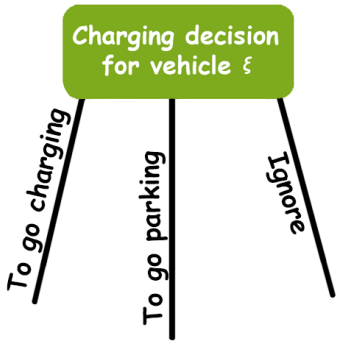


Figure 5-8 – Decision subtree at charging check.

During the charging cycle, the vehicle can accept the demand proposition and starts moving to the client location.

The leaf vertices contain the obtained reward, that is the number of clients served within their utility constraints. The first vague of pruning consists of the pruning the impossible branches because of previous actions. Thus, the construction of the final search tree requires the exhaustive search of all possible actions in the system. The complex and expanded nature of the proposed search tree does now allow us to propose similar methods for large networks. However, this method which provides an exact solution, requires a lot of memory and execution time.

5.6.2 Heuristical method for system optimum search

In big networks only heuristic methods of the system optimum search are possible. The proposed method in the case of complete information is the method of *"look with few steps forward"* and is as follows.

The system is fully centralised and for every demand chooses a vehicle that brings the minimal potential useless empty run and delay during next s steps. The step for the vehicle here is the number of served clients. Consider the example in Figure 5-9 with 2 steps forward memory.

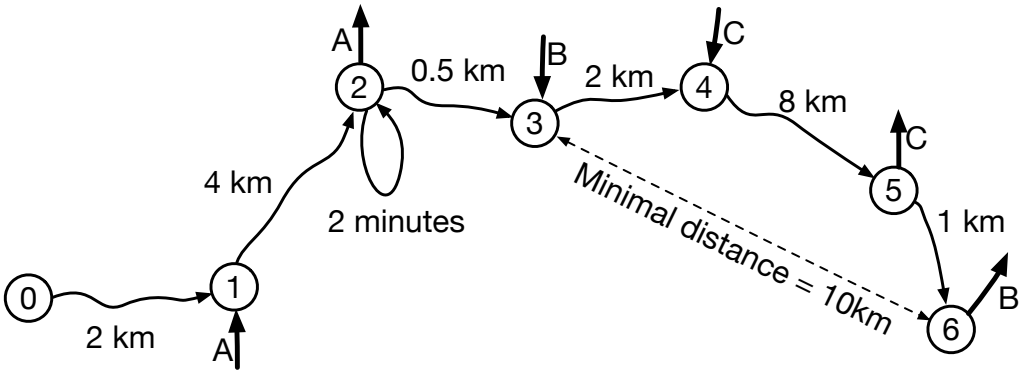


Figure 5-9 – Virtual path for vehicle.

Let us have the demand in Vertex 1. If we assign the vehicle located in Vertex 0, the trajectory that maximises the ratio of useful travelled client-distance to the total travelled distance will be the following: move from 0 to 1 and pick-up client A, depose it at Vertex 2, then move to Vertex 3, wait 2 minutes in order to pick-up client B, and move to Vertex 4 to pick up client C for a shared trip with B. Finally move to Vertex 5 to depose client C and afterwards to move to Vertex 6 to depose client B. Thus, the useful travelled client-distance is 4 kilometres for client A, 10 kilometres for client B (because the detour included in ride-sharing is not part of useful distance) and 8 kilometres for client

C with the total of 22 kilometres. However, the total travelled distance is 17.5 kilometres with delay of 2 minutes caused by client waiting. Summarising the eventual distance the vehicle would drive with average speed on adjacent to this edge to real travelled distance we get unified travelled distance. Thus, the unified total travelled distance is 18.5 kilometres and the required effective ratio is $\frac{22}{18.5}$. However, it is not necessary that after picking up the client in Vertex 1 the vehicle will move as shown in Figure 5-9. The route can be recalculated during next steps based on actions of other vehicles.

This effective ratio replaces the learning coefficients by the exact value of the "vertex promising", thus, of effective ratio described above. Thus, the learning coefficients do not need to be recalculated, which can provide better results in big networks, minimising the probabilistic nature of actions.

5.7 Conclusion

In this chapter we considered a set of general method of electric autonomous taxi system management. These methods use both reinforcement learning techniques and algorithmic heuristical techniques. The proposed RL methods are united in 3 main groups: microlevel, mesolevel and macrolevel learning. The microlevel model consists of vehicle-based RL model with classical representation of vehicles as agents and station-based RL method, where the memory of vehicles depends on their position. The mesolevel supposes the learning based on city graph clustering. The macrolevel supposes shared learning information and solving of optimisation subproblems based on matchings, that aim to diminish the empty runs in order to provide more energy save solutions. As for methods for the complete information case, in this chapter was proposed a method based on search tree pruning, and, for big networks, a method of "look with few steps forward" .

Figure 5-10 summarises dependencies between developed machine learning methods and algorithmic solutions discussed in Chapter 4. For example, one of necessary methods for station-based centralised method is the station-based decision model which is a reinforcement learning technique, which belongs to the class of machine learning problems.

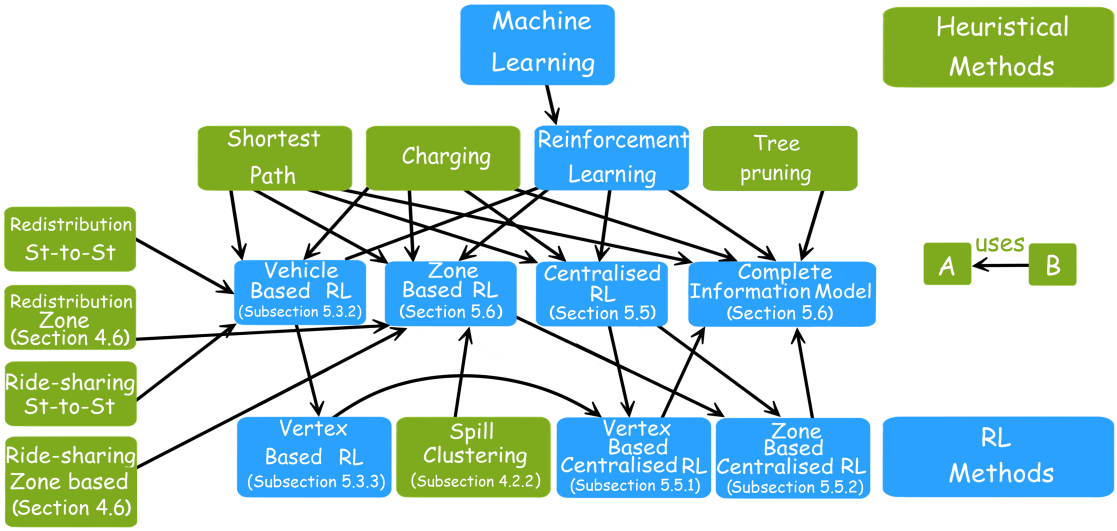


Figure 5-10 – Scheme of modules dependencies

To sum up, this chapter provides final solution techniques for aTaxi system management. The set of proposed solutions is designed for use in various transport networks. For example, for small networks we propose station-based RL model and for big ones, the zone-based RL model. The centralised models can show better performance in terms of execution time. Thus, for real-time systems and in the cases of obstacles in the network (for example, the vehicle looses the signal in the tunnel) a no centralised one is preferable, but for numerical experiments and long-term analysis, the centralised learning one is used. The complete information optimisation is used to analyse the system performance a-posteriori.

Thus, it cannot be used in real-time, but it remains important for the performance analysis.

All the described methods, in order to diminish the state and action spaces, can use different heuristics at different levels. In the numerical results Chapter 7 we provide explicit analysis of the system performance under different models and justify that RL model with applied heuristics shows better convergence as well as better results in long time.

Chapter 6

Evaluation scenarii

6.1 Introduction

This chapter is devoted to the evaluation scenarii description, including client, vehicle and network parameters. The chapter starts with a new method of relevant road graph construction.

Nowadays in transport modelling, research and development teams begin to spend more time working on complex practical problems such as mass transit arrival predictions [177], classification of vehicles [250], travel behaviour analysis [236], congestion prediction and analysis [190], demand patterning [59], etc. Using GPS data to make progress in solving these problems is beneficial. Lately, GPS data are becoming available in the form of big data sets [142, 138, 140]. This suggests searching for viable working methods aimed at making good use of such sets. A common existing approach is to use a road graph, usually created from OpenStreetMap (OSM) [1] data. This approach demands significant attention to the resulting graph, because the result is often incomplete: it lacks detailed information about speed constraints, one-way roads, closed sections, and unused sections.

In order for these data to be useful, the sets need to be applied to a road graph that is correct, consistent and has suitable edge lengths and travel times. Successfully solving the problem of travel time prediction (an important subproblem in traffic macroscopic simulation) is impossible without the statistical information on real travel time being a part of the road graph [57, 225]. To reflect reality, data on real movements should be used.

The classic city street network primal approach [215] uses a directed weighted graph to model the city [79]. Within this graph described in Subsection 3.3.1, the set of vertices represents road intersections with road end points, and the set of arcs represents relevant road segments. The weights of arcs are time-dependent functions. Such a graph ought to be constructed from real-world data. In order to obtain the graph structure for a network (without weights based on congestion), the OSM [118] data is mostly used. There are two major problems with a graph obtained using OSM: firstly, such a graph does not take into account real road structure and secondly, it can also have some invalid data. In addition, when the graph is obtained from a map, its arcs cannot contain any information about congestion and real speed.

Another way to construct a city graph is to use another data source, such as GPS vehicle data. Nowadays, such data sets become broadly available both as open data sets and as commercial data sets. Open data are mostly provided by taxi services (for example, taxi movements in New York [141], Chicago [137], Porto [143] etc.) for research purposes. The quality of GPS trajectory data obtained in the urban environment may be low. This usually happens due to the satellite signal loss by different obstructions: tunnels, buildings, trees. According to Wang et al. [276], the quality of the real-life taxi GPS trajectory data becomes a significant challenge for road map generation algorithms.

Another challenge in road map generation is that the density of GPS trajectories is unevenly distributed. In the literature, there exist several different approaches to construct street maps from

[1] The OpenStreetMap, commonly abbreviated to "OSM", is a non-commercial project aimed at making a complete and free geographical world map.

GPS tracking data only. These can be organized into the following categories [6]: Point clustering (this includes k-means algorithms and KDE[2] [37]) [241, 128], incremental track insertion [48], and intersection linking [147].

To properly synthesize information from two different sources into one usable graph, special algorithms are needed. Map-matching algorithms are currently a part of any land vehicle navigation system. The aim of such algorithms is to enhance the accuracy of vehicle positioning. According to Quddus [222], map-matching algorithms are divided into three groups: geometric (using of the geometric information of the digital road network by considering only the shape of the links but not the way they are connected to each other [114, 283]), topological (make use of the geometry of the links as well as the connectivity and contiguity of the links) and advanced (using mixed methods as well as real movement predictions [221, 9, 224]). The methods of matching GPS coordinates to the graph are based generally on the physical distance projection methods [221]. This requires either introducing non-straight line nature of the edges, or dividing edges into small parts to provide the necessary geometry.

The graphs obtained from OSM data are mostly used as benchmarks for methods of obtaining the road map from the GPS tracking data [241]. According to our knowledge, there are no open source utilities or published methods aiming at obtaining the real weighted map using both the pre-constructed graph based on OSM-like data and the GPS tracking data sets.

Taking these facts into account, it appears promising and necessary for the thesis purposes to work on the construction of a graph from both OSM and GPS data sets. The goal is to obtain a graph that combines strong points of both data types: the exact geographical structure of OSM data and the real movements of vehicles of GPS data. Processing these data leads to deriving smaller road graphs, which can be then used as a basis for traffic macroscopic simulation models and thus, for solving problems of traffic optimisation such as predicting travel times or scheduling.

This chapter starts with Section 6.2, where the focus is made on the relevant road graph construction. The method proposed in this chapter uses Dijkstra-based auxiliary methods to find the optimal trajectory through the GPS data based arcs. The resulting process of building a corresponding path inside the graph can show good enough prediction of the real vehicles trajectories. Polyline data and demand, demand patterns, distribution of origins and destinations are investigated. The resulting graph, constructed from both OSM and GPS polyline data, allows obtaining velocity patterns on the arcs, which are the necessary elements for aTaxi system management. The results obtained in Section 6.2 are used in order to effectively construct the evaluation scenarii.

Section 6.3 starts with the description of client generation with all their parameters including the generation time distribution, the time window constraints, the utilities and the ride-sharing and walking preferences. The taxi fleet parameters and initial distributions are described in Section 6.4. Section 6.5 describes the networks chosen for the electrical aTaxi system evaluation and is dedicated to the evaluation scenarii chosen in this thesis, including real-life and artificial networks with different sizes, demand patterns and vehicle number and characteristics.

6.2 Construction of a relevant consistent weighted city road graph

We first present in Subsection 6.2.1 the methodology we use to produce a relevant graph using the GPS trip data set and OSM data. Subsection 6.2.2 describes initial GPS data and some results of the analysis, which helps to show what kind of data the work is based on. Subsection 6.2.3 is dedicated to obtaining and preprocessing the raw graph based on real data. It describes the process of its optimisation using the GPS trip data set and depicts results of the optimized graph investigation of the road network of the city of Porto, Portugal.

The results obtained below are used in order to effectively construct the evaluation scenarii.

[2] Kernel Density Estimation

6.2.1 Methodology

Network representation

The network representation in this chapter is similar to the one used in Subsection 3.3.1. Thus, the network is represented by a directed weighted graph $G = G(V, A)$ with double weighted arcs $a = (init_a, ter_a, distance_a, speed_a(t))$.

Remind, that the theoretical time to pass the arc (street) $time_a(t)$ can be calculated as ratio of $distance_a$ and $speed_a(t)$. This value depends on the physical length and of the congestion on the road during the time period (rush hour, etc.). The weight of every arc is to be obtained for workday/holiday separately. In transportation modelling, it is a good practice to assume that the speed depends on the number of vehicles passing a street in given moment [29]. However, in this work the weight of the arc is considered independent of the number of the autonomous taxis on it because this number is significantly smaller than the number of common vehicles.

The travel time prediction for every arc is based on the real movement statistics and is obtained by the clustering methods. The common clusters are congested traffic, normal traffic and empty road.

GPS data analysis

The use of the GPS data for positioning is common, but GPS coordinates are not exact, especially for a moving vehicle. A high density of the GPS locations for one moving vehicle can help to restore the real vehicle movement trajectory on the map with good reliability. For example, modern third-generation personal navigation assistants mostly record one GPS tick per 1-15 seconds. In most of the taxi GPS data sets available ([142, 138, 140]), one line of data represents one complete taxi ride with the GPS coordinate polyline (with the coordinates taken every 10-15 seconds) and the departure time of the ride. It can also be composed of multiple attributes, such as the taxi ID, the call origin, the number of clients onboard, etc. These taxis use mobile data terminals installed onboard the vehicles.

The demand for a taxi as well as the congestion situation in the city are not constant during the day.

As far as all the data have the geographical coordinates as GPS points, the distance can be found using specific formulas for geodesic distance G_{xy}. This one can be defined as the shortest path between two points on the surface of the earth (without considering rivers and mountains). Let (ϕ_x, μ_x) and (ϕ_y, μ_y) denote the latitude and longitude of points x and y, respectively. The Vincenty inverse formula assumes Earth to be an ellipsoid (WGS84 coordinates) to calculate the geodesic distance. This is more accurate than the great circle distance methods [270]. In this chapter we use the Chang et al. [59] variation of the Vincenty formula, where the radius of the Earth is assumed to be 6372.795 km.

$$G_{xy} = R \cdot \arctan\left(\frac{\sqrt{(\cos\phi_y \sin(\mu_x - \mu_y))^2 + (\sin\phi_y \cos\phi_x - \sin\phi_x \cos\phi_y \cos(\mu_x - \mu_y))^2}}{\sin\phi_y \sin\phi_x + \cos\phi_x \cos\phi_y \cos(\mu_x - \mu_y)}\right). \quad (6.2.1)$$

OSM data preprocessing

In order to obtain the graph structure of the network (without congestion-based weights), the OpenStreetMap data are used. We chose the OsmToRoadGraph project [1] to convert OSM files into a graph of the real map. This project is an open-source tool for such a conversion.

Data preprocessing includes deleting lines with speeds more than 130 km/h and lines with missing data, as well as transforming the initial CSV data set into a more usable format.

The first problem of these data arises from the fact that every (even small) road turn found in the OSM data yields a vertex of degree 2 in the resulting graph, and this influences basic graph characteristics. Slow updates on map information are the second problem — they lead to inaccuracies in the graph: closed roads, no information about one-way roads, etc. The graph created by these scripts has lots of additional information and is rather extensive. The script proposes some in-built solutions to simplify the graph, but with a lot of information loss and some mistakes in specific cases.

To avoid the unnecessarily complex structure of the resulting graph, we provide some modifications to the OsmToRoadGraph script to simplify the structure. These modifications are implemented as additional modules: an updated module of graph contraction to fix a few of bugs at OsmRoadToGraph stage and a module for removing sinks and sources from the graph (we consider the graph to be strictly connected).

Mixed method of city graph construction

Here we describe the method we use to obtain the weighted (by the speeds and distances) graph from both OSM data and GPS data sets. The method consists of the preprocessing, the processing and the clustering of the obtained speeds on arcs.

The GPS data preprocessing is made the following way.

First, the data is converted to pickle format. Second all the polylines with missing data are deleted. Finally, all the obtained polylines with a speed greater than 130 km/h are deleted because of official speed limits inside a city, and thus we consider these trips as not suitable to study.

Data processing takes place as follows.

First of all, we try to fit each GPS polyline on the graph. The method used in Rahmani et al [224] has been changed by taking the nearest vertices of the graph instead of the nearest edges by real geometrical projection, which makes the calculations much faster without losing significant information (an opportunity of having a lot of polyline data).

After, we compute vehicle speeds for each edge. We add special labels for start and end points. These points do not belong to the initial graph, they are the start and end points of the polyline. We add them to the graph connected to their nearest neighbours with link speed of 10km/h.

The speed calculation for an arc is done by calculating the average time to cross this arc and the average speed between two projections of consecutive (with a 15 seconds difference) GPS coordinates. The arc lengths are calculated as a geodesic distance (see Formula 6.2.1).

In order to cluster the data based on the rush hours and the common traffic conditions, Regression Trees method is used. Predicting the real passing time of the arcs is possible with the resulting data.

6.2.2 Porto taxi data set analysis

To analyse the existence of patterns in GPS data, we consider the data set of the city of Porto, Portugal, which was the training set of the Kaggle [139] competition ECML/PKDD 15: Taxi Trajectory Prediction (I) [143]. This data bundle provided our research with a data set containing trip information on the trajectories for all the 442 taxis running in the city of Porto from 01/07/2013 to 30/06/2014. This data set contains over 1.7 million data points.

The data set owners categorize each ride as a taxi central based "A", a stand-based "B", or a non-taxi central based "C".

"A" trips are dispatched by the central operators. There are 364770 calls of this type in total. The call-time histogram of trips count with 5-minutes step in Figure 6-1(b) shows a pattern close to the classic passenger demand distribution with the morning rush-hour and two small evening rush-hours, which seems to be explained well by Porto common work hours.

"B" trips are demanded directly from a taxi driver at a specific stand. There are 817881 calls of this type in total. The histogram in Figure 6-2(a) reveals a situation similar to type A calls, but with brief pick up periods at the train arrival moments, exhibiting strong scheduled patterns.

"C" otherwise (a trip demanded on a random street). There are 528019 calls of this type in total. In Figure 6-2(b), features are likely tied to the movement outside the public transport schedule and have a strong peak between midnight and 7:00 am. It can be assumed that these are mostly not work, but leisure trips: the peak is at 4:00 am, the closing hour for most of the bars on Fridays and Saturdays (based on TripAdvisor).

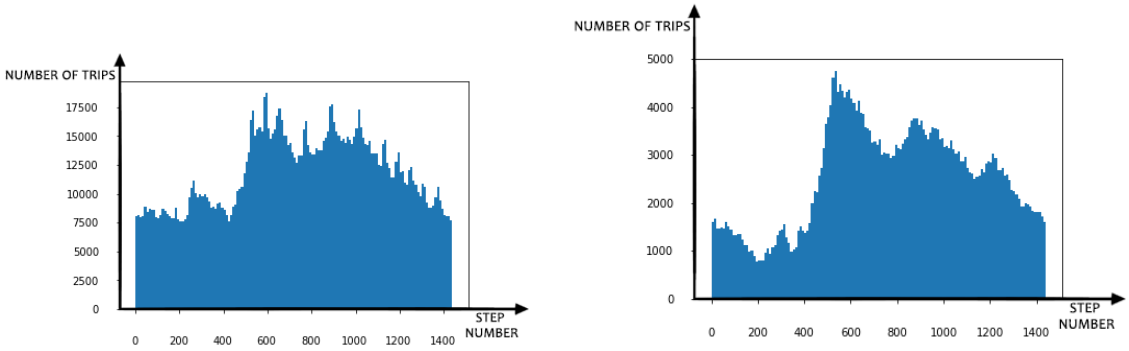


Figure 6-1 – Histogram of total taxi call numbers for (a) all calls types; (b) type A calls.

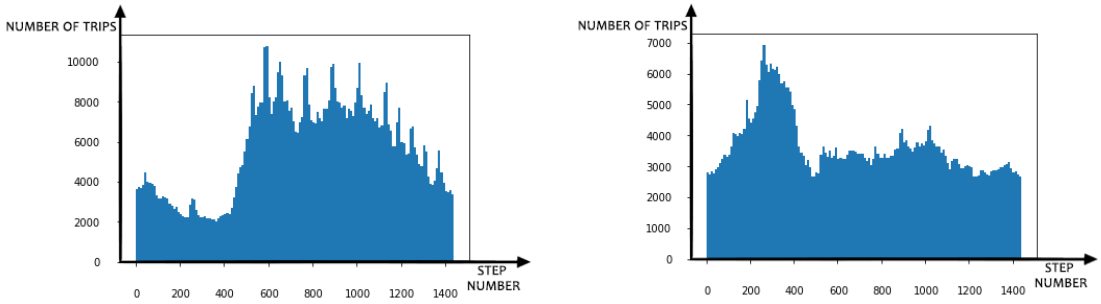


Figure 6-2 – Histogram of total taxi calls numbers for (a) type B calls; (b) type C calls.

Some of the basic statistics for these data seem interesting. The histogram in Figure 6-1a) (total number of calls) shows bigger demand during the working hours with slight peaks depending on train scheduling.

The origin and destination points for May Saturdays for type C calls (the "heat map") are shown in Figure 6-3. Here, there is a tendency for origins to be at the parks and at the bar streets. Destination points are distributed more uniformly.

Type A calls data for Tuesday mornings (Figure 6-4) suggest a uniform distribution of the origin points with destination points exhibiting strong peaks at the central train and bus stations.

These statistics may give a better understanding of the initial data. The other major step, processing these data, is the subject of the next section.

6.2.3 Building the Porto network

The methodology described in Section 6.2.1 can be applied to almost every city with available GPS data of vehicle movements. The OSM data are widely available. As it was mentioned above, the available formats vary depending on data type. As it is shown in Figure 6-5 (a) and (b), the OSM data have smaller granularity, whereas the GPS data have more complex structure and are less exact.

Figure 6-3 – OD point heat map for May Saturdays for type C calls (a) Origins; (b) Destinations.

Figure 6-4 – OD point heat map for Tuesdays from 7:00 am to 10:00 am for type A calls (a) Origins; (b) Destinations.

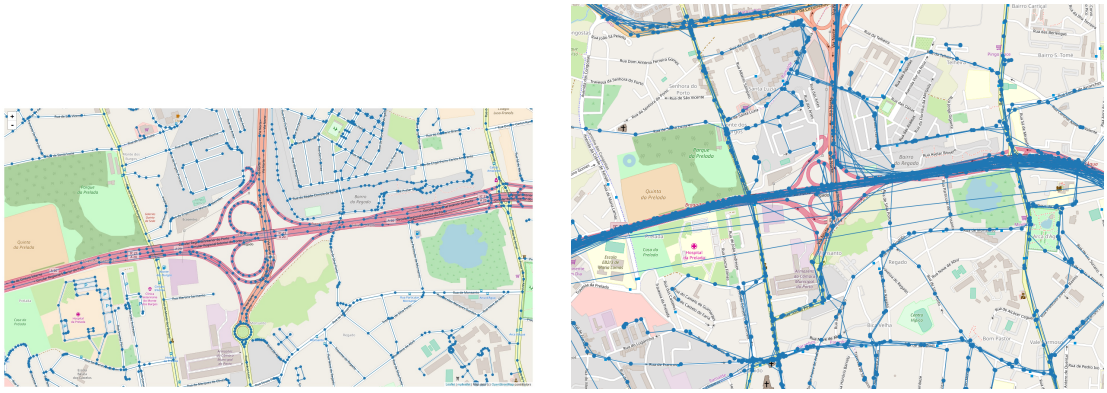


Figure 6-5 – a) OpenStreetMap graph export of data b) GPS data of Porto taxi movements.

Clustering of speeds on each arc

The method above has been applied to the analysis of the Porto network using the OSM data of the city area and the taxi movement data set described in Subsection 6.2.2. The real speed of the congested traffic has a lot of variations (shown in Figure 6-6). The clustering methods based on Regression Trees are applied on every arc of the graph of the system. Despite the fact that, in considered data set, the number of taxis is much smaller than the total number of vehicles in the city, there is a well-noticeable correlation between the number of taxis and their speeds.

Graph characteristics of the obtained Porto network

Having obtained some confidence in applicability of our approach, we can now study the graph closely. In order to be able to simulate the traffic of an arbitrary city, it is necessary to understand the general characteristics and traffic graphs formation patterns. Traditionally, a full road network graph is used to characterise the city. More practical and scientific values can be found in the graph after its simplification, associated with the removal of unused or closed roads and parallel arcs, often denoting different lanes and so on. Such a simplified graph obtained by the algorithms described above is shown in Figure 6-7.

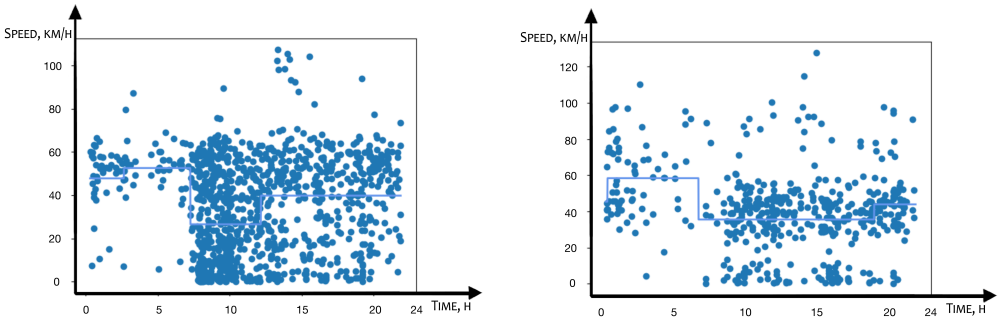


Figure 6-6 – Speed clustering depending on the time of the day.

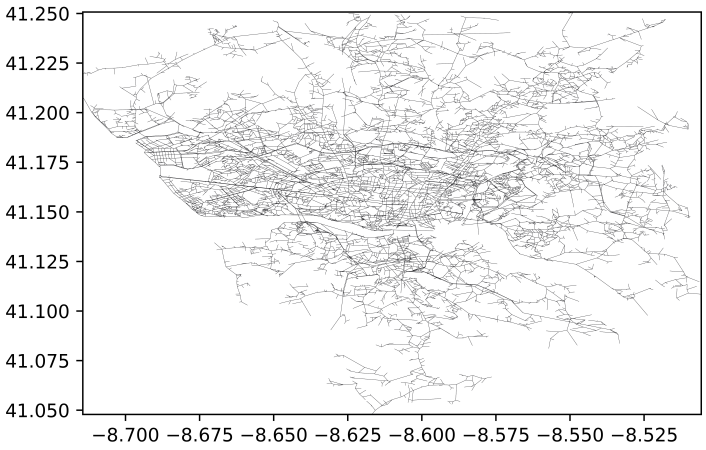


Figure 6-7 – The reduced graph of Porto area.

The number of vertices $|V|$ of this reduced graph with deleted geometry and parallel edges is 14154 while the number of edges $|E|$ is equal to 36580.

6.3 Client generation

The heterogeneity of passengers preferences is a fact that is used in various fields of transport research. The time delay of taxi booking by phone can vary depending on the necessary time of the ride, the destination of the ride (for example, rides to airports have tendency to be booked in advance), personal preferences of the client, and not only. Data of Singapore taxi companies reveal that approximately 3,000 advance reservations are made during the daytime (from 8:00 a.m. to 6:00 p.m. on a typical weekday) [271]. More precisely, in Singapore nearly 40 advance-booking demands are made every five minutes during the peak hours (from 7:00 to 9:30 in the morning and from 16:30 to 19:00 in the evening) [273]. When booking long trips, for example, train or plane tickets, the system has an easy tool to calibrate this time distribution by price changing and refund strategies [252, 226]. When booking a taxi, such an option is not available, primarily because of less predicted nature of trips. To recalibrate the taxi demand, in some taxi systems the advance reservation is defined as a service that should be fulfilled at most 30 minutes later [272].

The **client generation time** in this work is assumed to be predefined by the existing statistics or to be modelled by Poisson distribution if not provided in data set. The **desired waiting time** when calling the taxi by phone is supposed to have normal distribution, which is justified by the fact that clients in the system can be generally expressed as independent units that are similar in expectancy and variance. To sum up the observations above, the common time delay for calling taxi by phone is supposed to be not negative number and to have normal distribution with the expectancy of 15 minutes. The standard deviation of advance booking time is supposed to be 30 minutes [293].

When the information about the number of phone calls is not available, the statistics based on Porto data (see Subsection 6.2.3) are applied and the percentage of such a taxi calls during the simulation is supposed to be 20 percents.

The **ideal pick-up time** t_p^i for calls that require immediate service is supposed to be equal to t_p^0, whereas for calls made in advance, $t_p^i = \frac{2 \cdot t_p^0 + t_p^1}{3}$.

The **time window size** TW_p can satisfy one of the following conditions for different scenarii. In the case, referred to as "infinity time window", the time window is not limited. In "constant time window" case it is a constant of a given number of minutes. The last case, referred to as "variable time window", assumes normal distribution. The minimal possible value is supposed to be 1 minute.

Readiness to share of clients can have different distributions for different taxi systems. We consider 3 different types of sharing in the system. A "not shared", which represents individual taxis. A "partially shared" which is ride-shared economical-class taxi, and where a certain percent of clients is ready to share. In "fully shared" system, every client is ready to share its ride.

The **readiness to walk** parameter is modelled the following way. In "no walking" case, clients require service from their current station. In "constant walking" case, this distance is a constant of given number of meters, for example, 150 meters [305].

Number of passengers in one group has a relevant impact on the system, either because groups tend to spend more time in coordinating and boarding a taxi or a set of taxis, or because they also influent the number of taxis that are required. The passenger group size can differ a lot depending on the origin point. The statistics for airport origins [67] and destinations [55] show prevalence of group trips. However, according to our knowledge, the common trip statistics in not available in any open data source, therefore the statistics from [67] are chosen for the characterisation of the passenger group size. In this work, we consider 2 cases of client size distribution. First case, referred to as "single travellers", considers constant group size of 1 passenger. The "variable group size" is distributed unevenly. Table 6.1 provides the probability distribution we consider.

In the case of non-linear stochastic utilities, the **clients utility constraints** u_p^{min} are supposed to be distributed normally with expectancy of 0.75 and standard deviation of 0.5, limiting this value

Table 6.1 – The probability of group size distribution

Size of group	1 passenger	2 passengers	3 passengers	4 passengers
Probability	0.37	0.42	0.15	0.06

between 0.5 and 1.

The flexibilities of clients can be summarised in Table 6.2.

Table 6.2 – The flexibilities of client characteristics

	High	Medium	Low
Walking time	30 minutes	10 minutes	3 minutes
Walking distance	450 metres	150 metres	50 metres
Call time	3 hours	15 minutes	0 minutes
Ride-sharing	100 percents	20 to 60 percents	0 percents

In the evaluation chapter 7, we consider the **four main client types** described in Table 6.3. The distributions of clients of different types are individual for different evaluation scenarii and are described in the scenarii descriptions (Section 6.5).

These client types can represent different social, economical or behavioural preferences of customers.

6.4 Taxi fleet generation

In the model the capacity of the vehicles is a constant $D_\xi = 4$ passengers.

The maximal travelled distance on one finite battery charge for all the vehicles in the system is supposed to be $D_\xi^{max} = 483$ kilometres (300 miles), and the charging time from empty to full battery $Ch_\xi = 1$ hour. These numbers are characteristics of the Tesla Model S (2019) and rapid charge slot [181]. In the case of infinite vehicle battery, widely discussed in the literature for DRP problems, the maximal vehicle distance is not limited.

Vehicles are supposed to be generated uniformly in charging stations along the city, and fully charged at the start of the simulation. Moreover, the fleet-size is fixed for every simulation case.

Therefore, in the evaluation scenarii we provide the analysis of two cases based on vehicle battery characteristics: finite vehicle battery and infinite vehicle battery.

Table 6.3 – The types of clients

	Type 1	Type 2	Type 3	Type 4
Waiting time	30 minutes	10 minutes	10 minutes	3 minutes
Walking distance	450 metres	150 metres	50 metres	50 metres
Call time	0 minutes	15 minutes	0 minutes	3 hours
Ride-sharing	100 percents	20 percents or 60 percents	5 percents	5 percents or 60 percents
Ideal pick-up time	The start of time window	One third of time window	The start of time window	The start of time window
Size of group	Distributed from 1 to 4	Distributed from 1 to 4	Distributed from 1 to 4	Distributed from 1 to 4
Type of utility	Discrete	Discrete	Stochastic	Discrete

6.5 Scenarii generation

We consider five different network structures for the evaluation scenarii. The first 2 networks are artificial networks of simple structure, whereas the last 3 networks are real networks of different size and structure. These network structures are presented in the following subsections. These networks aim to evaluate discussed methods introduced in Chapters 4 and 5.

6.5.1 Ring network

One of the popular case studies of mathematical modelling of traffic flows is the modelling of traffic on the freeway, for example, on the ring road. One of the conveniences for modelling circular motion is its isolation, which allows considering the behaviour of the model in the long term [162]. The ring network models infinite feedback and can never crash. The simplest model to consider would be a one directional ring motorway. The simulation makes sense in such a model, because there are similar freeways in the world, for example, the Watford Ring Road in Watford, B4160 (Redditch Ringway) in Redditch in the UK, the Charleroi Ring in the city of Charleroi in Belgium, and many others.

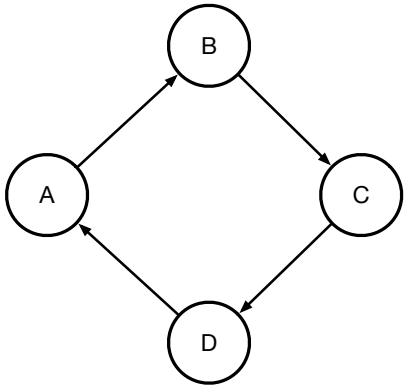


Figure 6-8 – The ring network.

The case study of simple ring network is shown in Figure 6-8. All the arcs have length of 2 kilometres, and all the speeds on arcs are considered to be constant of 30 km/h. Therefore, the time to pass one arc is 4 minutes.

The unique charging station, which is at the same time parking station, is located in vertex A and has probability of empty slot equal to 1.

This network was chosen to compare different methods of ride-sharing and charging.

At the start of simulation there is a unique vehicle functioning in station A and it is totally charged.

The demand in the system is predefined and clients are arriving the following way:

- In all moments equal to $0 + 33n$, where $n \in Z$, client a_n arrives to station A and is willing to reach station C;
- In all moments equal ro $4 + 33n$, where $n \in Z$, client b_n arrives to station B and is willing to reach station D;
- In all moments equal to $8 + 33n$, where $n \in Z$, client c_n arrives to station C and is willing to reach station B;
- In all moments equal to $8.5 + 33n$, where $n \in Z$, client d_n arrives to station C and is willing to reach station A;
- In all moments equal to $12.5 + 33n$, where $n \in Z$, client e_n arrives to station D and is willing to reach station B;
- In all moments equal to $21 + 33n$, where $n \in Z$, client f_n arrives to station B and is willing to reach station A.

In case of complete information, each client in the system can theoretically be served within its time windows as described in Appendix C.1.

The objective of the study of this scenario is to evaluate microlevel machine learning strategies and complete information strategies as well as heuristical methods of ride-sharing and empty vehicle management. In this scenario, the vehicle characteristic to study is finite/infinite vehicle battery and the influence of vehicle battery capacity on the system performance.

6.5.2 Grid network

In a lot of transportation problems, the modelling of grid network is one of important evaluation parts. Note that under a grid structured road network, as long as the trajectory of the trip falls within the defined rectangular boundary, its distance will always be the same as taxicab metric. Therefore, in such a type of network, the shortest distances are given as set of trajectories with the same Manhattan distance. Thus, choosing just the "shortest" path is not possible, especially when implementing the ride-sharing.

The network in this scenario is a square 2-way grid of 9 stations.

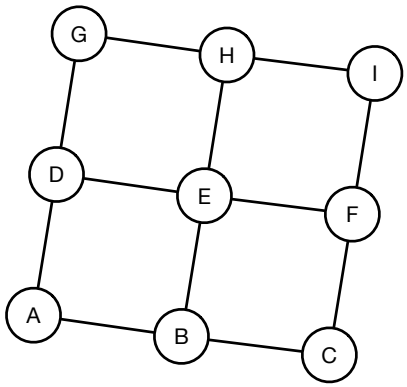


Figure 6-9 – The grid network.

The unique charging station, which is at the same time the parking station, is located in vertex E (Figure 6-9). It has a probability of empty slot equal to 1. The speeds on the arcs are constant equal to 30 km/h, and the sides of cells are equal to 2 kilometres. Thus, the time to pass one edge in any direction is equal to 4 minutes.

In the scenario of this network, the clients and their destinations are distributed uniformly. Thus 8 clients per time period are generated for every station, which makes in total 72 clients generated during simulation time period of 50 minutes. When all the clients leave the system, the simulation stops. Among these generated clients, 36 are generated at time 0, and 36 clients 10 minutes after the start of the simulation.

The fleet-size in this scenario is variable and is one of the characteristics to study. Because of relatively small simulation time, the charging strategies are not of interest to study in this scenario. Thus, the vehicles are supposed to have infinite battery. However, the main research interest here is the evaluation of the ride-sharing.

The objective of this study is to evaluate microlevel machine learning strategies, centralized machine learning strategies, complete information strategies, as well as methods of ride-sharing and empty vehicle management. For these methods we provide multiple test runs of the repeating simulation patterns.

6.5.3 Saclay network

The first real network based scenario is the network of Paris-Saclay, France (Figure 6-10). This area has been selected as the French test bed for testing and evaluating autonomous vehicle services.

The Paris-Saclay area contains the largest French university campus, housing the université Paris-Saclay, the science department of Orsay, ENS, Télécom Paris, as well as a number of national research

institutes (ENSTA, ENSAE, Technocentre Renault). In 2014, it comprised almost producing 15% of the total national research capacity. The area is being developed intensively (it is ont the largest public-private investment projects in Europe).

The proposed autonomous taxi network is displayed in Figure 6-10 and connects the main campuses to the Massy public transportation hub (SNCF trains, RER commuter trains). The extent is 10 km from Massy station to the furthest campus. The network contains 20 stations. The parts in red correspond to the current Bus Rapid Transit (BRT) network. For the aTaxi network we expanded this network with the parts in blue.

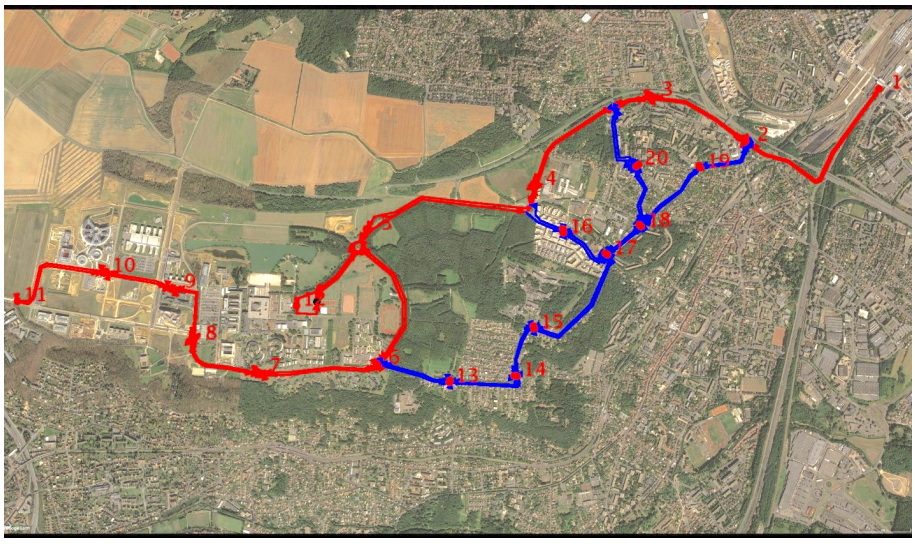


Figure 6-10 – Saclay network.

The speeds on arcs are considered to be constant and equal to 30 km/h. The charging stations are not included in the network. The simulation of this network assumes infinity vehicle batteries.

In this network, the demand patterns are based on real data. The peak hour passenger demand between the 20 stations (where one has no passenger demand) was obtained from the Île de France Transport Authority (STIF), via assignment in a regional model for Île de France in VISUM. We consider two main case studies. The first considers real demand patterns, and the second considers 50% of the original demand. The vehicles which supposed to have infinite battery are distributed uniformly at the start of the simulation. The number of vehicles varies depending on case study and one of the objectives of this scenario is to provide the fleet size analysis.

The objective of the study of this scenario is to evaluate heuristical methods of ride-sharing and empty vehicle management. Here, we consider not-shared and fully shared systems under no time windows case. Thus, we consider the classical setting of the empty vehicle redistribution problem. The main objective is to evaluate different methods of empty vehicle redistribution under different demand and fleet-size. The objective functions in this scenario are the client waiting times as well as the number of served clients.

6.5.4 Stockholm network

This simplified city network of the real case is provided as a shortest time matrix. We provide a system with 103 stations, that are metro stations of the city of Stockholm (Figure 6-11).

The maximal travel time in the shortest time origin-destination matrix is 46 minutes. The charging stations are not included in the network. The simulation using this network assumes infinity vehicle batteries.

The case study of Stockholm is generated based on the real-world taxi floating car data from 1,500 taxis operating in Stockholm, Sweden region. Each taxi reports its GPS coordinates, timestamp, and status about its occupancy rate. Frequency of these reports is on average every two minutes. There

Figure 6-11 – Stockholm metro stations.

is no special information about the true origin and destination of the trip. In order to infer them, the change in the status between two adjacent reports is used. The trip with client and R reports $r_1, ..., r_R$ is represented by binary vector $(0, 1, 1, ..., 1, 1, 0)$, where 1 means that at the time of report r_i the taxi is occupied and 0 otherwise. To represent the possible spatial space for true origin and destination, we consider them as the line or ordered vector of size two: (r_1, r_2) for the origin and (r_{R-1}, r_R) for the destination.

For the computational experiments we consider only the none empty trips during morning peak hours (06:00 - 10:00) on 1st April 2016 (see Figure 6-12(a) for the spatial density of all trip origins in the Stockholm region). In order to simplify the network for the purpose of the computational experiments, we aggregate trips with origin and destination in a radius of 2 kilometres in the surroundings of one of the 103 metro stations. Figure 6-12 (b-c) show magnitudes of demand (number of trips) and destinations aggregated to the metro stations. The magnitude of the selected trips that originate and terminate near the metro stations constructs the OD matrix. These data can be used for a stationary Poisson point process simulation as the trips themselves represent inputs for real demand case with real call times.

The expectancy of the total number of clients arriving to all the stations during the simulation period of 10000 seconds is 1451. Another simulation is to be provided by considering 50% of the original demand.

The number of vehicles is equal to 120 and they are distributed uniformly at the stations at simulation start. The vehicles are assumed to have infinite battery.

The objective of the study is to evaluate the heuristical methods of empty vehicle management. Here, we consider the not-shared system under conditions of infinite or fixed time windows. The walking is not included because the distance between every 2 stations is greater than the accepted limit by client. The main objective of this case study is to evaluate different heuristical methods of empty vehicle redistribution with and without time windows under classical EVR problem objectives such as the client waiting times (average and maximal) and the number of served clients.

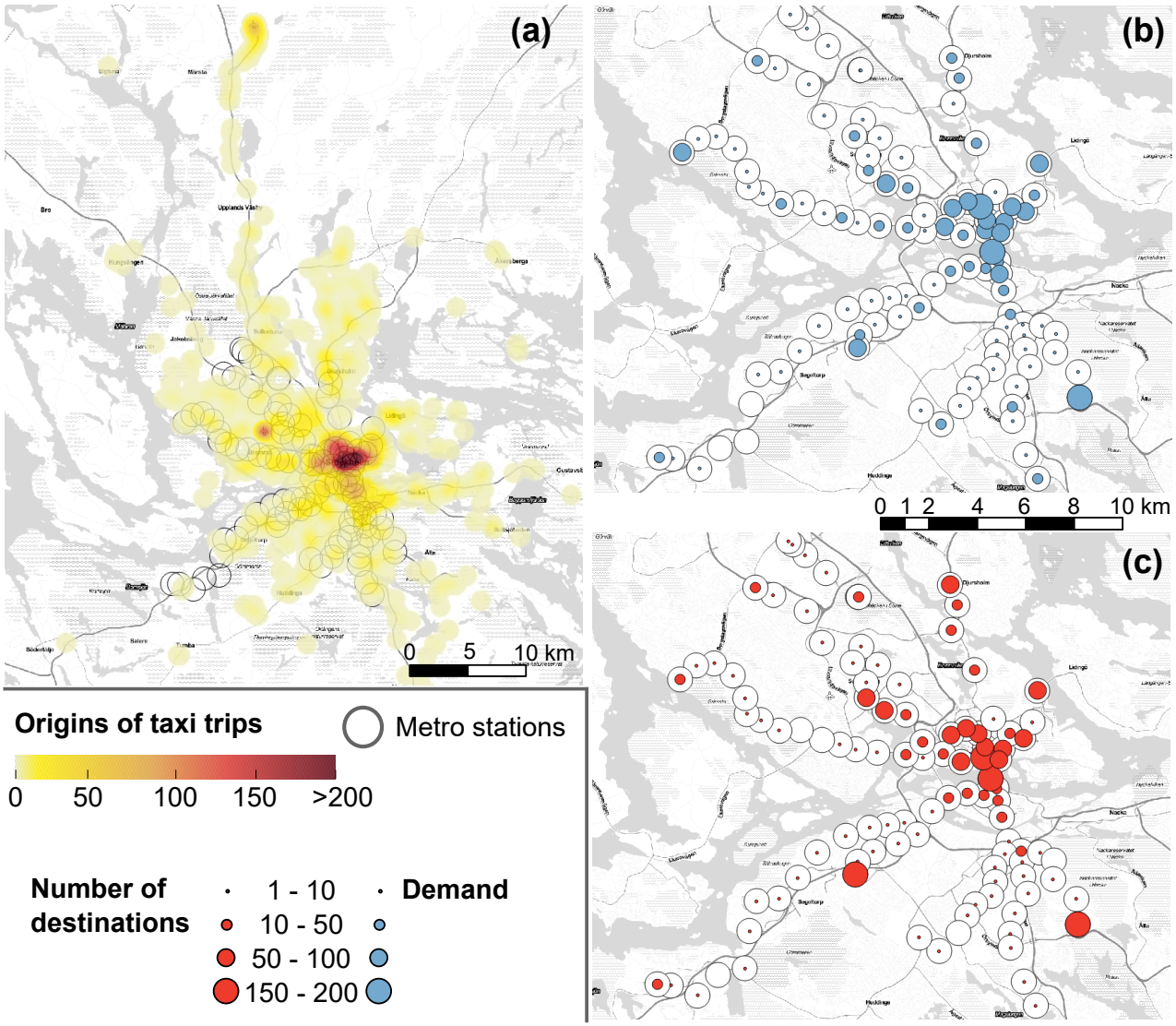


Figure 6-12 – Scenario visualisation: (a) spatial density of all trip origins before aggregation to metro stations, (b) demand aggregated across all metro stations and (c) destinations of the same trips.

6.5.5 Porto network

The biggest case study considered in this work is the Porto city network, obtained by methods described in Section 6.2. While in Subsection 6.2.3 the full network of the Porto area is obtained, as a case study we use the network of smaller region of the Porto city. In addition, most of the arcs that have less than 10 taxis passing during 1 year period are eliminated, taking into account the need for the graph to remain strongly connected. The number of vertices in the network is equal to 3705, and the number of arcs is 8913. The speeds on arcs in the given road graph are clustered for every day, whereas distances are fixed.

The number of parking stations is supposed to be constant and equal to 100. Generally, the maximal travelling distance between adjacent parking stations depends on the relevant clustering. The capacity of each station is supposed to be equal to 200. The other vehicles arrivals are supposed to be proportional to the client demand and to have a Poisson process nature. The parking times in public parking stations are supposed to be fixed and equal to 60 minutes. Based on the empty slot heuristics described in Subsection 4.3.1, the empty slot probability for every station, at any moment, can be calculated. Half of the parking places are supposed to be equipped by charging pylons. This

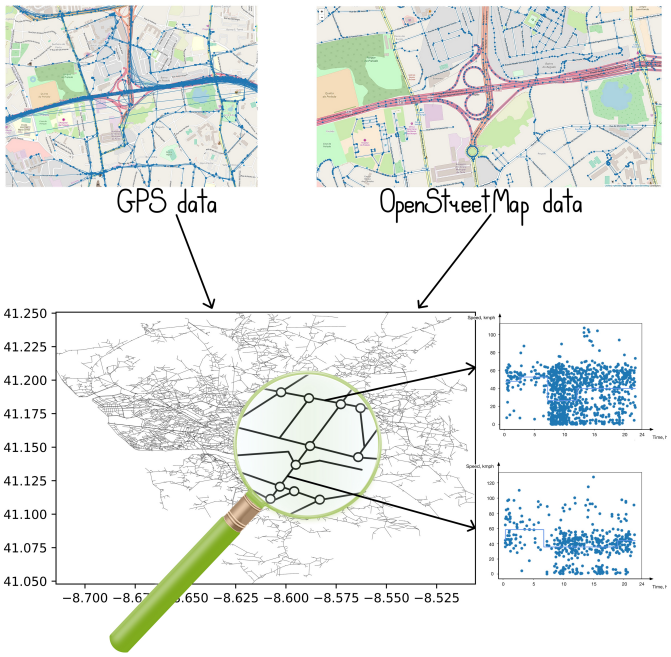


Figure 6-13 – The construction and analysis of a consistent weighted city road graph based on both GPS and GSM data.

charging stations are distributed uniformly through the parking stations.

We consider the case of real network structure and real client arrivals, based on this network.

For the proposed data set we consider that client calls of types “B” and “C” require immediately service, whenever the client calls of type “A” can be made in advance as described in Subsection 6.3. The client distribution of different types is also considered in the evaluation. The vehicles are assumed to have finite battery. As a training set for machine learning methods, we consider the demand data of 364 days from 00:00:01 01/07/2013 to 23:59:59 29/06/2014 with fixed taxi fleet of 442 vehicles. The number of trips during this time period was 1310196.

The study objectives of this scenario are to evaluate the mesolevel machine learning strategies, centralised machine learning strategies, complete information strategies, and the use of heuristical methods of ride-sharing and EVR at different levels.

The fleet-size in this scenario is variable and is one of the characteristics to study.

The chosen evaluation set is the demand data of 1 day from 00:00:01 to 23:59:59 of 30/06/2014 that was a workday (Monday), with a total of 3725 trips.

6.6 Conclusion

This chapter is devoted to the evaluation scenarii and case studies description.

In the beginning, this chapter describes methods for obtaining a consistent, real weighted road graph of a city, based on OpenStreetMap (OSM) network data and real vehicle movement GPS data sets. The proposed generic graph generation method is centred around finding a trajectory on the OSM graph close to the GPS data polyline of the trajectory, to create a graph which can be then used in computational models and in transport network analysis. The weight of every arc of such a graph has a clustered structure depending on the weekday and time of the day, as it is shown in Figure 6-13, which is necessary for the aTaxi management optimisation purposes. The resulting graph allows obtaining velocity patterns on the arcs to predict vehicle movements.

Chapter 7

Numerical evaluation of aTaxi systems

7.1 Introduction

This chapter is devoted to the evaluation of the aTaxi system using the five scenarii described in Chapter 6. Firstly in Section 7.2 we discuss the performance measures for the evaluation scenarii we are interested in.

The first part of this chapter, represented in Sections 7.3-7.6, is centred around the evaluation of the proposed methods in this thesis on small and medium networks, thus, on the first 4 scenarii. The size of these networks allows us to provide real-time optimisation at microlevel, thus, without using zone decomposition. Different network and demand structures allow us to study different strategies and methods of aTaxi management on them. For example, the one-way structure of the ring network does not allow us to study different strategies for ride-sharing, however, it allows us to compare results on shared and not shared vehicles. In contrast to one-way ring structure, the grid network allows the best simulation and comparison of different ride-sharing strategies.

The empty vehicle redistribution algorithms are applied on all the considered scenarii. The comparison between different empty vehicle redistribution algorithms with and without time windows is provided for Scenarii 3 and 4. In Scenarii 1 and 2 we use the STN redistribution method, and the BA method for the baseline. The fleet-size is studied in Scenarii 2 and 3. Scenario 3 also provides analysis of the system performances from the point of view of economics with introduction of the generalised cost.

The second part of this chapter is centred around the evaluation of proposed methods on real network, thus, on Scenario 5 of the Porto city.

In Section 7.7 we study the influence of developed methods on the system performances. We compare the obtained results on the real network under cases of deterministic and stochastic client preferences. The mesolevel based techniques are studied in order to evaluate scalability in this network. We describe the influence of the zone size on the obtained results. Under proposed methods, we compare RL techniques with deterministic ones. The obtained results are used to evaluate fleet-size under different client arrival densities and preferences.

We discuss the results in Section 7.8, and provide the conclusion.

The general information about the case studies are be summarized in Table 7.1.

Table 7.1 – The evaluation scerarii

		1.Ring	2.Grid	3.Saclay	4.Stockholm	5.Porto
Battery	Finite	**YES**	NO	NO	NO	**YES**
	Infinite	**YES**	**YES**	**YES**	**YES**	NO
Methods to evaluate	Comparison between EVR algorithms	NO	NO	**YES**	**YES**	NO
	Comparison between ride-sharing strategies	NO	**YES**	NO	NO	**YES**
	Usage of EVR in evaluation	**YES**	**YES**	**YES**	**YES**	**YES**
	Usage of ride-sharing in evaluation	**YES**	**YES**	**YES**	NO	**YES**
	Vehicle-based RL in evaluation	**YES**	**YES**	NO	NO	NO
	Station-based RL evaluation	NO	**YES**	NO	NO	NO
	Zone-based RL evaluation	NO	NO	NO	NO	**YES**
	Evaluation of centralised station-based methods	NO	**YES**	NO	NO	NO
	Evaluation of centralised zone-based methods	NO	NO	NO	NO	**YES**
	Evaluation of complete information	**YES**	**YES**	NO	NO	**YES**

7.2 Used methods and performance measures

Let us remind the strategies, which we will use and compare in this chapter.

The strategies for empty vehicle redistribution between stations are as in the Table 7.2:

Table 7.2 – Strategies

Reactive	Proactive
BA Basic Allocation	**SDR** Surplus/Deficit vehicle Redistribution
SNN Simple Nearest Neighbours	**IBR** Index-Based Redistribution
HNN Heuristic Nearest Neighbours	**IBRTL** Index Based Redistribution Time Limited
HNNTL Heuristic Nearest Neighbours Time Limited	
STN Send The Nearest	

The cited literature shows that reactive and proactive redistribution strategies have different strengths and weaknesses in various demand conditions and network configurations (see Subsection 4.4.3). In particular, it seems that simpler reactive strategies perform well in high unpredicted demand scenarii and dense networks, whereas lower or strong predicted demand scenarii and more stretched out network configurations may be better served by proactive strategies. In this chapter we therefore evaluate reactive strategies (SNN, HNN, HNNTL and STN) by themselves, as well as in combination with proactive strategies (SDR, IBR and IBRTL), to see if combining strategies may provide a service that performs well in both high and low demand situations. In the remainder of this section these redistribution algorithms and theirs combinations are noted as follows. For example, "SNN+SDR" notation means "Simple Nearest Neighbours combined with Surplus/Deficit vehicle Redistribution". The comparison of different EVR strategies is provided in Sections 7.5 and 7.6. For the baseline of EVR strategy we use Basic Allocation (BA), this means no redistribution of empty vehicles between the stations. If the contrary is not mentioned, for big networks with use of zone decomposition, we chose the combined methods for zone-based empty vehicle redistribution (Subsection 4.6), notably surplus/deficit zone-based EVR and Zone index-based EVR. For heuristics without zone decomposition, we use mixed strategy of STN and IBR. Hereinafter, we refer to cases with this empty vehicle redistribution as to "with EVR".

The empty vehicle redistribution algorithms are implemented in all the considered scenarii. The comparison between different empty vehicle redistribution algorithms with and without time windows is provided for Scenarii 3 and 4. In Scenarii 1 and 2 we use the STN redistribution strategy, and the BA method for the baseline.

When we use deterministic optimisation in order to provide the evaluation of the system performance, we consider the ride-sharing strategies, discussed in this work. If the contrary is not mentioned, we use the "Searching on the way" heuristic. In the cases where we compare ride-sharing heuristics, we refer to them as "SOTW" or "SOTA" (Searching in the area of the origin and/or destination). For methods with zone decomposition, we use method described in Subsection 4.5. In a baseline heuristic method, we use the "Searching on the way" heuristic. The abbreviations of these methods are as follows:

- **NRSH**: no ride-sharing;
- **SOTW**: with ride-sharing "Searching on the way" (see Subsection 2.3.2);
- **SOTA**: with ride-sharing "Searching in the area of the origin and/or destination" (see Subsection 2.3.2);
- **ZBRS**: with zone-based ride-sharing (see Section 4.5).

The ride-sharing is implemented in Scenarii 1, 2, 3 and 5. In Scenario 2 we compare between different methods of station-based ride-sharing strategies, whenever in other scenarii the use of "Searching on the way" 2.3.2 ride-sharing method is compared with non use of ride-sharing. In Scenario 5 we evaluate zone-based ride-sharing on the big network of Porto city.

7.2.1 Performance measures

The evaluation is provided using the main system performance measures, which correspond to the optimisation objectives formulated in Section 3.5. They are as follows:

- *Max wait (min)*: Maximal waiting time within served clients, in minutes;
- *Avg wait (min)*: Average waiting time within served clients, in minutes;
- *Not served (clients)*: Number of not served clients; the simulation time is proposed to be bigger than the maximal pick-up time of the last client. Thus all the clients leave the system before the end of the simulation;
- *Total run (km)*: Total run of the vehicles, in kilometres;
- *Empty run (km)*: Total empty run of the vehicles, in kilometres;
- *Occ run (km)*: Total occupied (non-empty) run of the vehicles, in kilometres;
- *Avg load (clients)* Average load of all the vehicles during all the runs, in terms of clients;
- *Avg queue (clients)*: Average length of queue of clients;

The stochastic nature of proposed algorithms and of proposed data sets reveals the need to additional measures and performance statistics. The number of necessary replications for simulation runs can be calculated from the desired size of the confidence intervals for the mean of the main outputs, as described in [62] and [49], where the required number of replications (N), given m initial replications, is given by the following Formula (7.2.1):

$$N(m) = \left(\frac{S(m) t_{m-1,1-\alpha/2}}{\overline{X}(m)\varepsilon} \right)^2 . \tag{7.2.1}$$

Where,

- $\overline{X}(m)$ is the mean μ from m simulation runs (samples);
- $S(m)$ is the standard deviation σ from m simulation runs;
- α is the level of significance;
- ε is the allowable error percentage of the estimate $\overline{X}(m)$. $\varepsilon = |\overline{X}(m) - \mu|/\mu$;
- $t_{m-1,1-\alpha/2}$ is the critical value of the two-tailed t-distribution at level of significance α, given $m-1$ degrees of freedom.

The same stochastic nature reveals the need for specific auxiliary performance measures, which correspond to iterative process of reinforcement learning methods, discussed in Chapter 5. Hereinafter, we will use the following abbreviations:

- TR_i: Total runs of the vehicles with i clients inside, in kilometres;
- $TS^{\bar{i}}$: Total number of served clients during i iterations;
- TS^i: Total number of served clients during ith iteration;
- $AvW^{\bar{i}}$: Average waiting time within served clients during i iterations, in minutes;
- AvW^i: Average waiting time within served clients during ith iteration, in minutes;
- $TR^{\bar{i}}$: Total runs of the vehicles during i iterations, in kilometres;
- TR^i: Total runs of the vehicles during ith iteration, in kilometres;
- $TER^{\bar{i}}$: Total empty runs of the vehicles during i iterations, in kilometres;
- TER^i: Total empty runs of the vehicles during ith iteration, in kilometres;
- ExT: The processor execution time of the total simulation, in seconds;
- ItC: The number of iterations to enter the converged state.

All the results in this chapter are obtained using 3.4GHz Intel 7600B processor with 48GB RAM.

7.3 Evaluation scenario 1: Ring network

This system is isolated and is the simplest model to show potential patterns of electrical aTaxi system management under different conditions.

In this case study, the ride-sharing methods based on combinatoric optimisation are implemented. We consider only the "Searching on the way" method (see Subsection 2.3.2), because of the structure

of the ring network. Moreover, we consider only client groups of size 1, with fixed time-window of 10 minutes and without calls in advance. The walking is not included in this scenario, because the distance between every pair of stations is big. As a remainder, the total number of clients in every iteration is equal to 6, and the simulations are iterated without pauses.

We consider here the vehicle-based learning model, because with only one vehicle in the system the station-based model is out of interest. As it appears, because of the structure of the network that is one-way ring road, the learning itself shows evident results: the vehicle will always take a client on its way, when there is a place and enough battery for the ride.

Let us consider the number of served clients after 100 iterations of the simulation process depending on the battery size (the charging speed is considered to be a constant) which gives 600 clients in total. The non-monotonous nature of the results in Figure 7-1 is explained by the fact, that small change in the battery capacity changes the delay in the charging station. The small additional delay in the charging station can provoke a cyclic impossibility to serve consecutively arrived clients. Thus, the results show clear evidence, that the ride-sharing improves the performances upon the case without it. The changes in the battery capacity can provide non-monotonous changes in the obtained results. Thus in the later sections the battery is either of constant finite capacity or infinite one.

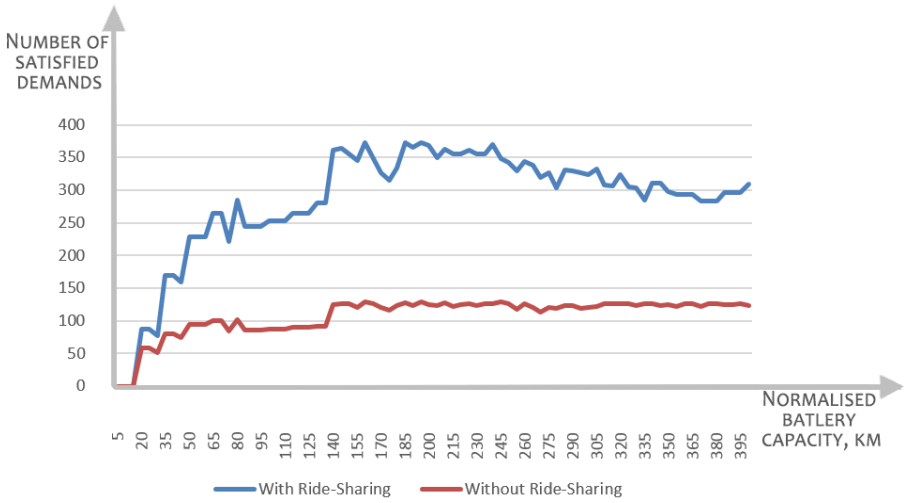


Figure 7-1 – Number of served clients with and without ride-sharing.

We have to insist on the fact that in the complete information case, the vehicle should wait for client in the station, producing additional delay for clients sharing their rides, which is not possible in the classical problem setting. The global optimum in this case is 0 served client under normalised battery capacity less than 8 kilometres (because the vehicle will not be able to return to the charging station if it starts to move), and all 600 clients will be served in the opposite case, as described in Appendix C.1.

7.4 Evaluation scenario 2: Grid network

We compare different methods of the electric aTaxi system management under different conditions on the grid network. The vehicle battery capacity is not of research interest in this case.

Remember, the total number of clients in every simulation is equal to 72. In this section we consider that the size of each client group to be equal to 1. All of them have time-windows between 10 and 30 minutes and without calls in advance. The walking is not included in this scenario, because the distance between every pair of stations is big. Ride-sharing in the evaluation cases below is either allowed for everybody or rejected. This is done in order to provide more exact and evident results of ride-sharing methods comparison.

7.4.1 Deterministic optimisation evaluation

We provide the evaluation of the system performances when using only deterministic optimisation, notably methods described in Section 4.4 and Subsection 2.3.2 for empty vehicle redistribution and ride-sharing, respectively. For the ride-sharing, we compare the results of the SOTW and SOTA heuristics as well as NRSH strategy. We consider no reinforcement learning in this subsection.

The methods we compare in this subsection are represented by combination of empty vehicle redistribution and ride-sharing strategies. The results of this comparison are summarised in Table 7.3.

Table 7.3 – Non-learning methods summary results.

	BA, NRSH	EVR, NRSH	BA, SOTA	EVR, SOTA	BA, SOTW	EVR, SOTW
Not served (clients)	52	49	34	**32**	34	**31**
Avg wait (min)	8.35	8.27	4.43	**4.33**	4.18	**4.11**
Max wait (min)	24	23.4	24	**22.1**	24	**23.2**
Total run (km)	78	100	64	**94**	66	**86**
Empty run (km)	0	16	0	**28**	0	**20**
TR_1 *(km)*	78	84	18	**20**	20	**20**
TR_2 *(km)*	0	0	16	**18**	12	**12**
TR_3 *(km)*	0	0	16	**16**	18	**18**
TR_4 *(km)*	0	0	14	**12**	16	**16**
Avg load (clients)	1	0.83	2.10	**1.62**	2.46	**1.89**

The results clearly show better performances with EVR and ride-sharing methods. The EVR implementation improves the performances in terms of the number of served clients, but this is also achieved by bigger number of empty runs. Thus, in all three evaluations with different ride-sharing strategies (NRSH, SOTA, SOTW), the average load becomes smaller than in simulation without EVR. Consequently, the EVR influences also the average waiting time which becomes slightly smaller. The ride-sharing method SOTW provides better results for the grid network, in terms of empty runs and energy consumption, then the SOTA heuristic. It can be explained by the considered network nature, where the areas of origin and destination are either limited by one station, provide not acceptable delay for clients, or are included in SOTW heuristic. Therefore, as a main ride-sharing strategy for the considered network we choose SOTW heuristic.

7.4.2 Station-based RL evaluation

The station-based reinforcement learning model can be used separately or, as described in Section 5.3, it may be accompanied by deterministic optimisation methods. In this subsection, we provide the evaluation of these options. However, the evaluation of the reinforcement learning model should include parameters, such as convergence characteristics and the its evolution with the time. In order to provide these data, the simulations were carried out as follows. One of the outputs of every simulation are the new values of learning coefficients, which are considered as inputs for a new learning iteration. Thus, the convergence in the case of repeated patterns can be shown clearly.

The optimal values for coefficient set $\{C^s\}$ (see Subsection 5.3.3) are obtained using differential evolution method with the objective of the maximisation of the total number of served clients after 100 iterations. Reinforcement learning model has a probabilistic nature, so a significant number of iterations and the significant deviations was reached. The numbers are shown below.

Table 7.4 summarises the results for the different learning coefficients with EVR+SOTW optimisation methods for a system with 9 vehicles. The format of the learning function (5.3.1) (see Subsection 5.3.2) is uniform on the coefficient set. Thus, the multiplication of every coefficient by a certain number does not change the value of the function. Therefore we provide in Table 7.4 the coefficient sets normalised to the total sum of 1.

Table 7.4 – Coefficient sets for station-based reinforcement learning.

	C_t	C_s	C_d	C_a	C_w	C_{CS}	C_{FD}	C_{FOD}	C_{FA}	C_{FW}
$\{C^s\}_1$	0.099	0.035	0.209	0.157	0.035	0.064	0.023	0.262	0.093	0.023
$\{C^s\}_2$	0.079	0.043	0.266	0.165	0.058	0.072	0.072	0.129	0.079	0.036
$\{C^s\}_3$	0.100	0.100	0.100	0.100	0.100	0.100	0.100	0.100	0.100	0.100

Table 7.5 – Learning results for station-based reinforcement learning.

	$\{C^s\}_1$	$\{C^s\}_2$	$\{C^s\}_3$
$\overline{TS^{25}}$	30.12	11.84	7.04
$\overline{TS^{100}}$	35.42	28.49	11.74
$\overline{TS^{400}}$	37.72	36.26	24.64
TS^2	8.1	5.3	5.1
$\overline{AvW^{400}}$	3.58	3.57	5.27
AvW^2	6.29	7.19	9.10
$\overline{TR^{400}}$	105.7	116.6	99.2
TR^2	84.7	85.1	68.4
ItC	12	68	533

Table 7.5 provides the main results we have obtained for the learning under different values of the sets. We can see, that set $\{C^s\}_1$, which was obtained using DE method, shows the best performances.

Figures 7-2a, 7-2b, 7-2c present progress results for three machine learning processes, corresponding to provided sets of coefficients. These figures confirm the results presented in Table 7.5 and highlight the non-constant nature of the converged state. Figure 7-2a represents progress for first set of tuning coefficients. The graphic shows a dependency between the number of served clients and the iteration number. Around 10th iteration the learning process starts oscillation around converged result of 36 served clients. Figure 7-2b provides the results for another set $\{C^s\}_2$, which becomes oscillating only around the 50th iteration. The results obtained under set $\{C^s\}_3$, as shown in Figure 7-2c, become converged after iteration 533. The final converged results in all three cases oscillate around 36 served clients.

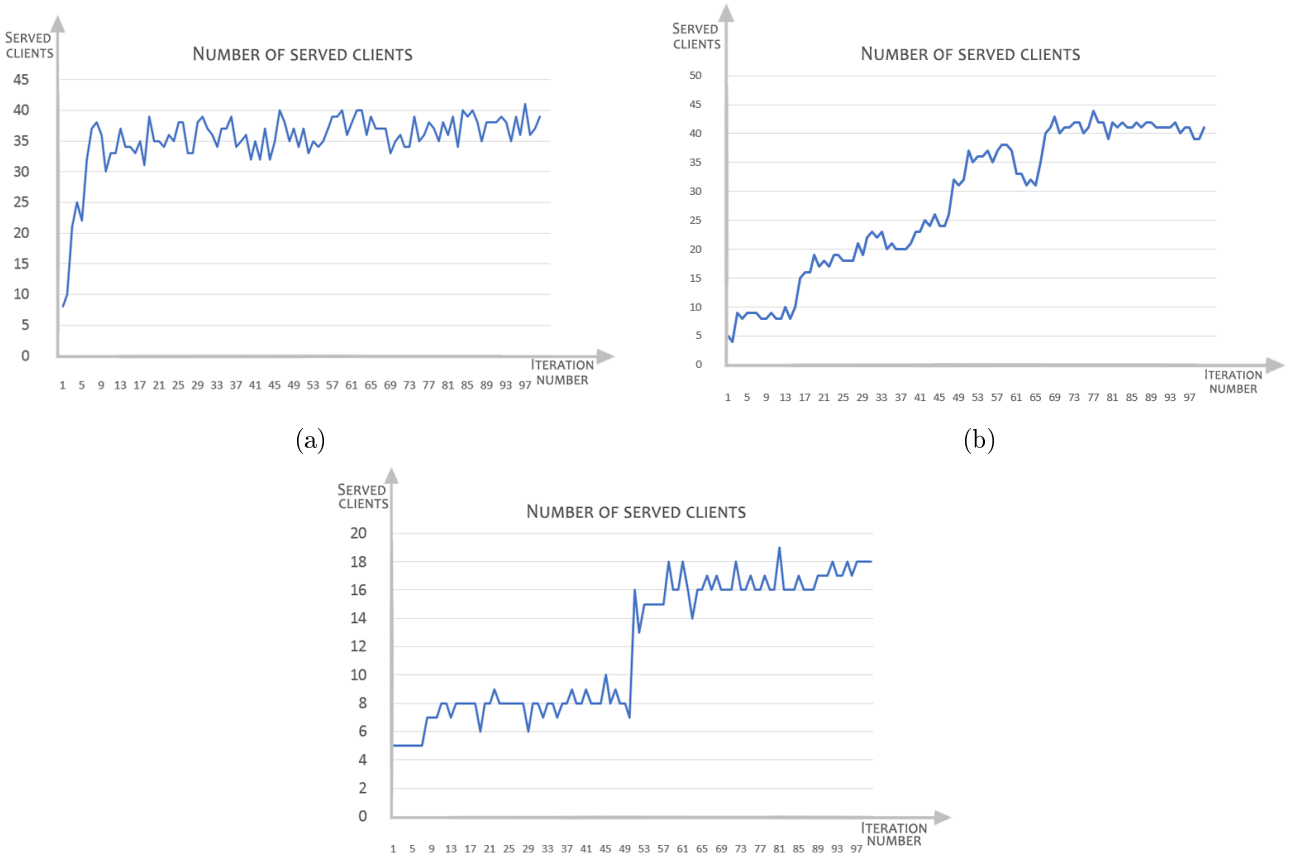


Figure 7-2 – Learning progress: Number of satisfied clients during one iteration for (a) $\{C^s\}_1$; (b) $\{C^s\}_2$; (c) $\{C^s\}_3$.

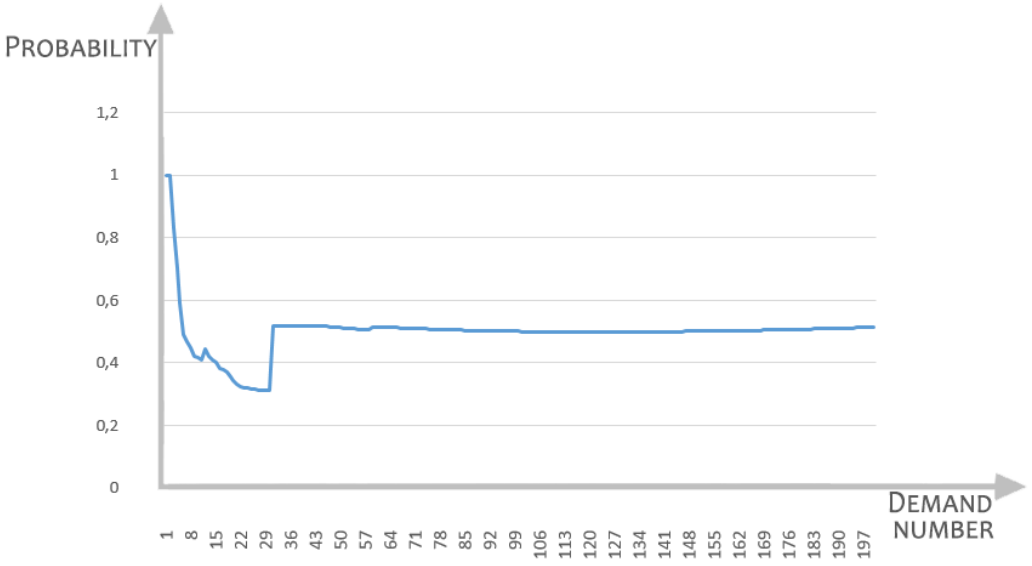


Figure 7-3 – Learning progress for the probability to accept a demand.

Results of function G (5.3.1) depend on the learning process and change with each call (see Section 5.3.2). Figure 7-3 shows progress of function G per call, using set $\{C^s\}_1$, for the probability that the system accepts a demand from client who is located at station E and desires to reach destination

station D. The results show that the convergence of the calculated probability is achieved around the 33th served demand between considered stations E and D. The client data set of this scenario provided 3 calls per iteration between these stations. Thus, the convergence is achieved around the 11th iteration.

7.4.3 Vehicle-based RL evaluation

The optimal values for coefficients $\{C^s\}$ in the vehicle-based RL evaluation were obtained by differential evolution method with the objective of maximising the total number of served clients after fixed number of iterations as described in Subsection 7.4.2. The obtained optimal values of these coefficients are provided in Table 7.6.

Table 7.6 – Optimal coefficients set for vehicle-based reinforcement learning.

	C_t	C_s	C_d	C_a	C_w	C_{CS}	C_{FD}	C_{FOD}	C_{FA}	C_{FW}
$\{C^s\}_1$	0.103	0.034	0.212	0.123	0.027	0.116	0.014	0.192	0.110	0.068

Table 7.7 provides the performance results using the obtained optimal set $\{C^s\}_1$. The obtained results are slightly less promising than the ones obtained for the station-based reinforcement learning. Thus can be explained by the slower convergence nature, because the learning is distributed over vehicles.

Table 7.7 – Learning results for vehicle-based reinforcement learning.

	$\overline{TS^{400}}$	TS^2	$\overline{AvW^{400}}$	AvW^2	$\overline{TR^{400}}$	TR^2
$\{C^s\}_1$	34.07	5.21	3.64	6.88	105.3	80.2

After obtaining the values for the learning coefficients, we compare the performances of the model with fixed values. We provide the evaluation of the average performances in every iteration as a result of 100 runs for each 400 iterations sequence. Table 7.8 summarises the results.

For the column names we use the following abbreviations:

- $VBMNH^i$: Vehicle-based method without combinatorial optimisation methods (neither ride-sharing nor EVR), the averaged results for the ith iteration;
- $VBMH^i$: Vehicle-based method with combinatorial optimisation methods (both ride-sharing and EVR), the averaged results for the ith iteration;

Table 7.8 – Vehicle-based model comparison for cases with and without combinatorial optimisation methods.

	$VBMNH^2$	$VBMNH^{400}$	$VBMH^2$	$VBMH^{400}$
Not served (clients)	67.14	34.18	65.34	30.45
Avg wait (min)	6.29	3.40	6.07	3.27
Max wait (min)	28.1	27.3	24.3	22.9
Total run (km)	17.9	105.7	16.1	105.9
Empty run (km)	0	0	0.6	2.2
TR_1 *(km)*	17.9	105.7	5.6	30.1
TR_2 *(km)*	0	0	3.8	31.9
TR_3 *(km)*	0	0	4.1	27.3
TR_4 *(km)*	0	0	2.0	14.5
Avg load (clients)	1.0	1.0	2.08	2.21

The number of iterations before convergence for $VBMNH$ and $VBMH$ methods are equal to 340 ± 20 and 295 ± 20, respectively. Thus, the results in Table 7.8 are presented for 400th iteration.

These results show, that the methods with enabled combinatorial optimisation methods provide better results. This is due to more effective use of vehicle fleet, including the enabled ride-sharing and empty vehicle redistribution. The fleet is used more effectively, and the average load is just one of its characteristics.

7.4.4 Centralised station-based RL evaluation

The centralisation of the system aims to achieve better results than in a non centralised one. In this subsection, we evaluate the performances of the centralised station-based reinforcement learning.

As in previous subsections, we use the differential evolution method to obtain the optimal values for coefficients $\{C^s\}$. The obtained set is provided in Table 7.9.

Table 7.9 – Optimal coefficients set for centralised station-based reinforcement learning.

	C_t	C_s	C_d	C_a	C_w	C_{CS}	C_{FD}	C_{FOD}	C_{FA}	C_{FW}
$\{C^s\}_1$	0.119	0.013	0.319	0.094	0.006	0.069	0.025	0.225	0.088	0.044

After obtaining the values for the learning coefficients, we evaluate the performances of the centralised station-based reinforcement learning under these coefficients. In Table 7.10 we provide the performance results averaged for 100 runs for 400 iteration sequence. We provide the results for the case without combinatorial optimisation methods. The results for enabled combinatorial optimisation methods are provided later in the fleet size evaluation Subsection 7.4.6. The obtained converged results outperform, in terms of the mean number of served clients, the results obtained for both vehicle-based and station-based methods. This can be explained by faster convergence due to the sharing of all the learning information as well as to the re-matchings which aim to reduce unsuccessful runs. Therefore, the fleet tends to be used more effectively.

Table 7.10 – Learning results for centralised station-based reinforcement learning.

	$\overline{TS^{400}}$	TS^2	$\overline{AvW^{400}}$	AvW^2	$\overline{TR^{400}}$	TR^2
$\{C^s\}_1$	38.12	7.0	3.57	6.79	110.7	90.0

7.4.5 Complete information

For the case of complete information in grid network, we use the method described in Subsection 5.6.1. In this evaluation case, we do not consider charging strategies, supposing infinite battery. The decision tree is constructed using 72 demands in the system. Each of them allocates relevant number of pick-up time decisions based on the client type and the time window. There are no charging decisions in the system, thus, the only decision vertices are the demand ones.

For the same client distribution, as above, with everybody accepting to share their ride, in the global optimum case, the statistics for each of the 9 vehicles are presented in Table 7.11. They show, that optimal results are possible only when occupied vehicles delay at certain stations in order to meet new demands.

Table 7.11 – Vehicle movement statistics in optimal strategy.

	Empty run (km)	TR_1 *(km)*	TR_2 *(km)*	TR_3 *(km)*	TR_4 *(km)*	*Avg load (clients)*	*Delay on vertices (min)*	*Served (clients)*
Vehicle №1		2	4	2	8	3		11
Vehicle №2		2		2	6	3.2		9
Vehicle №3		2	4	6	2	2.57		8
Vehicle №4		4	6		4	2.29		9
Vehicle №5		4	2	4	2	2.33	6	7
Vehicle №6	2	2	2	4		1.8		6
Vehicle №7		4	6	2		1.83		8
Vehicle №8	2	4	4	2		1.5	2	5
Vehicle №9		2	4		4	2.6	2	6

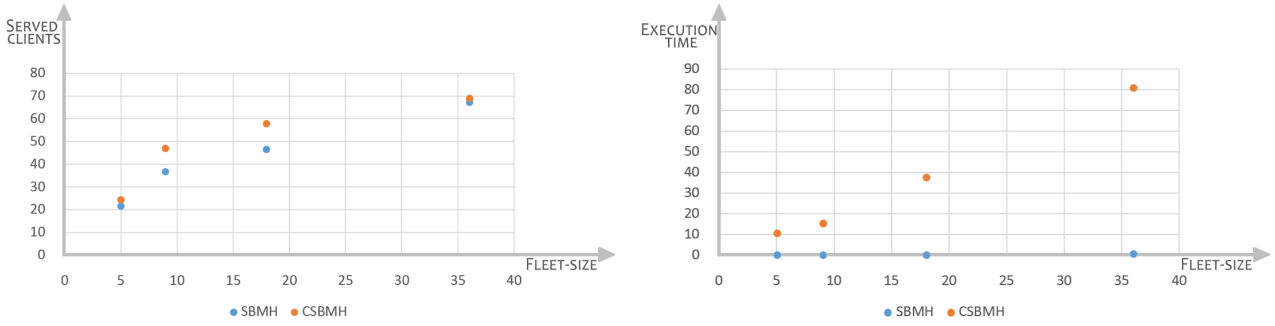
The results in Table 7.12 outperform all the provided above results in terms of the number of served clients. However, the average waiting time within served clients shows bigger value than in converged states for all the reinforcement learning methods. This can be explained by the fact that some of the clients which quitted the system in reinforcement learning simulations have waited longer and therefore were not counted. Nevertheless, the considered situation is not possible without prior knowledge, thus it is not achievable by any online method.

Table 7.12 – Optimal results for the complete information case.

	Not served (clients)	*Avg wait (min)*	*Max wait (min)*	*Total run (km)*	*Empty run (km)*	*Avg load (clients)*
Offline optimisation	3	4.35	12	110	4	2.36

7.4.6 Fleet-size evaluation

In this section we investigate variation of taxi-fleet and discuss the most promising methods among the investigated above. The system performances in previous subsections show, that the most promising within not-centralised methods is the station-based reinforcement learning method with combinatorial optimisation methods. As a centralised method, we choose the station-based centralised method with combinatorial optimisation methods.



(a) (b)

Figure 7-4 – Dependency between fleet-size and (a) number of served clients; (b) program execution time, seconds.

Figure 7-4a shows dependency between the number of served clients and the fleet-size. In both cases, with and without centralisation, the function is increasing with an asymptote of 72 served

clients (their totality). Under small number of vehicles (5) in the system, the sharing possibilities are consistent, as well as the re-matching ones. Thus, the number of served clients in both cases are almost similar. These numbers are also close to each other in case of big fleet-size (36 vehicles). This can be explained by their proximity to the possible maximum. Nevertheless, under fleet-size of 9 and 18 vehicles the difference is more significant (20-25 %).

Figure 7-4b shows dependencies between program execution time and fleet size. The execution time for this scenario grows nearly linear with fleet size in centralised system and increases less by 7.6 times when fleet size increases by 7.2 times. However, the nature of the grow can differ based on the demand distribution and fleet size. The re-matching complexity grows as a function of the number of idle vehicles and destination points in polynomial nature, and the frequency of re-matching possibility appearance grows in unpredictable way, depending on the client arrival intensity and the vehicle positions.

7.4.7 Summary

In this section we evaluate different optimisation methods on the artificial grid network. These methods are vehicle-based reinforcement learning, station-based reinforcement learning with and without centralisation, and complete information exact method. The gap between online and offline solutions is significant. However, to obtain offline optimal trajectories it is necessary to use delays on certain stations, that is impossible in online methods. With online methods, the optimal solutions can be obtained using combined methods which consist of reinforcement learning and combinatorial optimisation methods together. Within methods with combinatorial optimisation methods, the performances can be ordered as vehicle-based, station-based and centralised station-based RL methods, respectively. The station-based method outperforms the vehicle-based one because of its faster convergence. However, under centralised learning there is a possibility to achieve better results. Nevertheless, the nature of dependency between execution processor time and the number of vehicles under centralised learning reveals that under real-size networks and fleets the optimisation process can exceed the time needed for real-time simulation, or require specific computational resources. Thus, disregarding possible better results in terms of the number of satisfied clients, the proposed basis for optimisation in real systems of medium size is the station-based reinforcement learning with combinatorial optimisation methods.

7.5 Evaluation scenario 3: Saclay network

In this section we compare the algorithms presented in Section 4.4 applied to Saclay Network, Paris-Saclay, France. These algorithms were implemented as part of the event-based microscopic simulation model VIPSIM, developed by VEDECOM, France. This simulation tool of mobility service networks aims to define, test and evaluate operation strategies embedding all the relevant economic factors, including the quality of service for users.

The main performance results in this section are the client waiting time and the queue length. The results are presented for both the peak hour demand and in case of less traffic. We will consider the demand equal to half of the peak hour as a reduced demand. Hereinafter we refer to it as off-peak one.

Formula 7.2.1 (Section 7.2) was applied to the average and maximal client waiting times. Given initial replications $m = 100$, $\alpha = 0.05$ and $\varepsilon = 0.1$, the required number of replications is $N = 29.30$ for the average client waiting time and $N = 29.80$ for the maximal client waiting time. Thus, the 100 initial replication runs were sufficient to provide reliable results. The results presented below are an average over 100 simulation runs.

7.5.1 Evaluation on the greedy mixed algorithms

We deal with the mixed vehicle redistribution algorithms on the network, given a fixed fleet size of 100 vehicles. At every decision moment (every fixed time period) we choose between the method with fixed probabilities, bigger for reactive and smaller for proactive strategies. The greedy mixing is provided

Table 7.13 – Evaluation results: rush hour demand.

Strategy	BA	SNN	SNN+SDR	SNN+IBR	HNN	HNN+SDR	HNN+IBR
Max wait (min)	52	14	13.5	**13**	30	25	23
Avg wait (min)	31	3,3	3.2	**2.4**	7	6.2	4.7
Avg queue (clients)	872	31	30	**21**	95	72	57
Empty run (km)	0	73.4	72.6	**109.4**	74.6	72.9	130.7
Occ run (km)	8.7	172.5	182.1	**205.6**	169.8	173.3	200.0

by choosing the closest vehicle to the client with the highest priority. It was simulated 10,000 seconds of model time to predict the system performance in long terms.

Case 1: Rush Hour Demand

In Table 7.13 and Figure 7-5 the main results for the peak hour traffic are presented. In the BA strategy the maximum client waiting time is 52 minutes, the average waiting time is 31 minutes and at the end of the simulation there is on average a queue of 872 clients that have not been served. This is expected as there is no redistribution in this strategy, which is reflected in 0 minutes of empty run time for vehicles and 262 minutes of occupied vehicle run time, since vehicles are never relocated to match demand.

The simple nearest neighbour strategy SNN improves the performance on all counts, reducing the maximum waiting time to 14 minutes, the average to 3.3 minutes and the queue length at the end of the simulation to 31 clients. The empty run time goes up to 2202 minutes since now vehicles are relocated to match demand, and the occupied run time increases to 5174 minutes, since many more clients are transported.

The strategy where SNN is combined with the SDR, improves somewhat on the SNN, reducing the maximal waiting time to 13.5 minutes, average waiting time to 3.2 minutes, and the queue length to 30 clients. The empty run time is slightly improved to 2187 minutes and the occupied run time goes up to 5462 minutes.

Strategy SNN+IBR improves significantly upon the SNN and SNN+SDR strategies, leading to the lowest values across the table for maximal waiting time (13 minutes), average waiting time (2.4 minutes) and average queue length at the end of the simulation (21 clients). However this comes at the cost of increased empty run time (3282 minutes). The larger number of clients served is also reflected in the larger number of occupied vehicle runs, 6169 minutes.

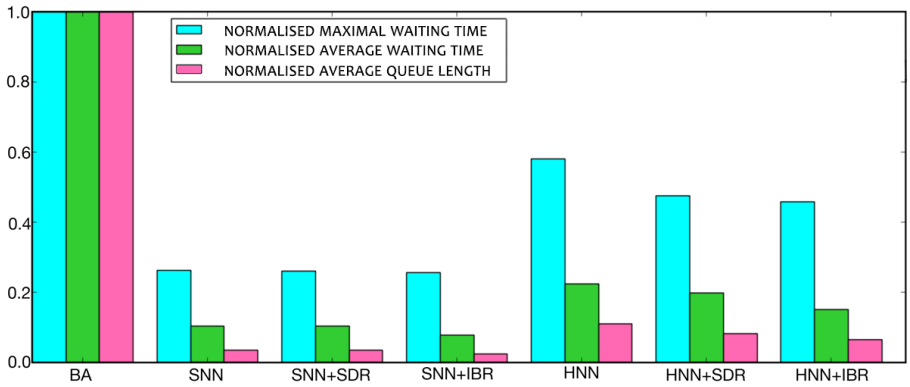


Figure 7-5 – Rush hour traffic. Comparison between different strategies.

Strategies HNN, HNN+SDR and HNN+IBR produce less favourable maximal and average client waiting times, and queue length results than the SNN based strategy. However they lead to comparable empty run and occupied run times.

SNN+IBR mixed strategies: In this paragraph we look in more details at different mixes of the IBR and SNN algorithms. We apply the combination of the two strategies of variable intensities (See Subsection 4.4.3). At every simulation step, one of the redistributions is called, so it can be called few times per second (distributed uniformly, as described in Subsection 4.4.3). In Figure 7-7, we plot as performance measure the queue length at the end of the simulation (Z-axis) and in Figure 7-6 the average client waiting time (Z-axis) for the strategies with different IBR (X-axis) and SNN intensities (Y-axis) in times/second.

In Figure 7-7, the IBR=0 (i.e. Y-Z plane) shows the performance of SNN solely redistribution and the SNN=0 (i.e. X-Z plane) shows the performance of IBR solely redistribution. We note that with increased intensities, the average waiting time and the queue length decrease rapidly for both strategies, when applied separately. When the strategies are combined, the result is 25% shorter in average waiting time than when the strategies are applied separately. In addition the performance surface (in X-Y-Z) is smooth indicating that all combinations of SNN and IBR intensities lead to smooth variations in the average waiting time. From this we can conclude that the mixed strategy is robust with regard to the intensity parameter variations.

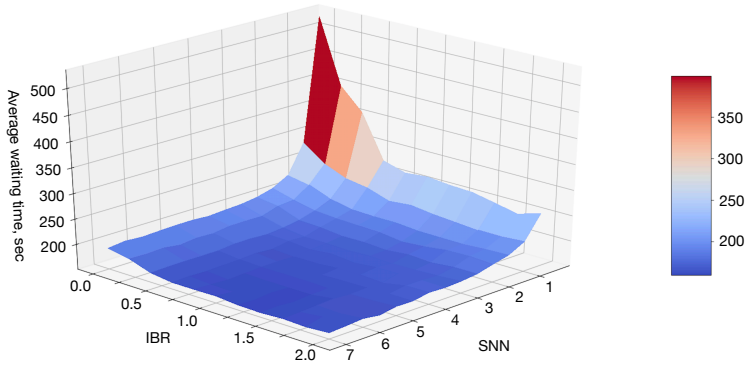


Figure 7-6 – Rush hour traffic: IBR-axis: Intensity of the IBR, times/second; SNN-axis: Intensity of the SNN, times/second; Z-axis: Average waiting time, seconds.

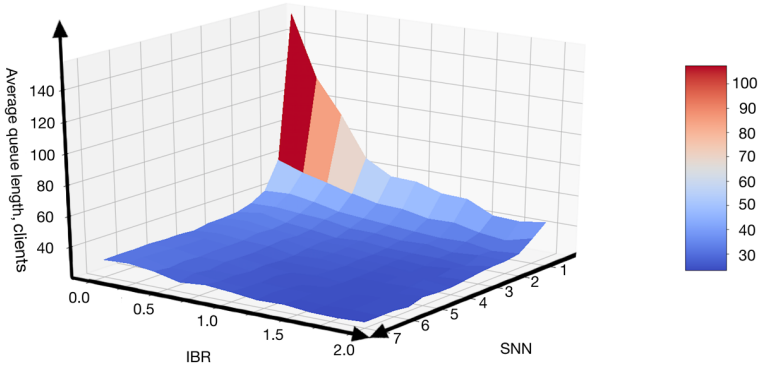


Figure 7-7 – Rush hour traffic: IBR-axis: Intensity of the IBR, times/second; SNN-axis: Intensity of the SNN, times/second; Z-axis: Average queue length at the end of simulation, clients.

In Figure 7-8, for the best performing SNN+IBR strategy mix (with intensity parameters SNN=6 and IBR=1.2), we show the normalised (by division to value network in total) average and maximum waiting times, as well as the normalised queue lengths at the end of the simulation. The results are shown for every station in the system separately. For four stations (4, 7, 8, 9) there is no demand in the morning peak hour (these are work places on the campus), and therefore no queues and waiting

Table 7.14 – Evaluation results: off-peak traffic.

Strategy	BA	SNN	SNN+SDR	SNN+IBR	HNN	HNN+SDR	HNN+IBR
Max wait (min)	60	8,4	7.9	**5.1**	10.7	9.6	6.2
Avg wait (min)	20	0.78	0.77	**0.25**	1.13	1.10	0.36
Avg queue (clients)	872	31	30	**21**	95	72	57
Empty run (km)	0	22.3	22.2	**221.8**	23.0	23.6	210.9
Occ run (km)	20.3	91.6	88.7	**88.4**	87.6	87.8	88.2

times. For stations 6 and 18 the most significant queues lengths are reported, cumulating for almost 45 percent of the queue lengths in the network. On the other hand, the distributions of the average and maximal waiting times are much more uniform across the stations.

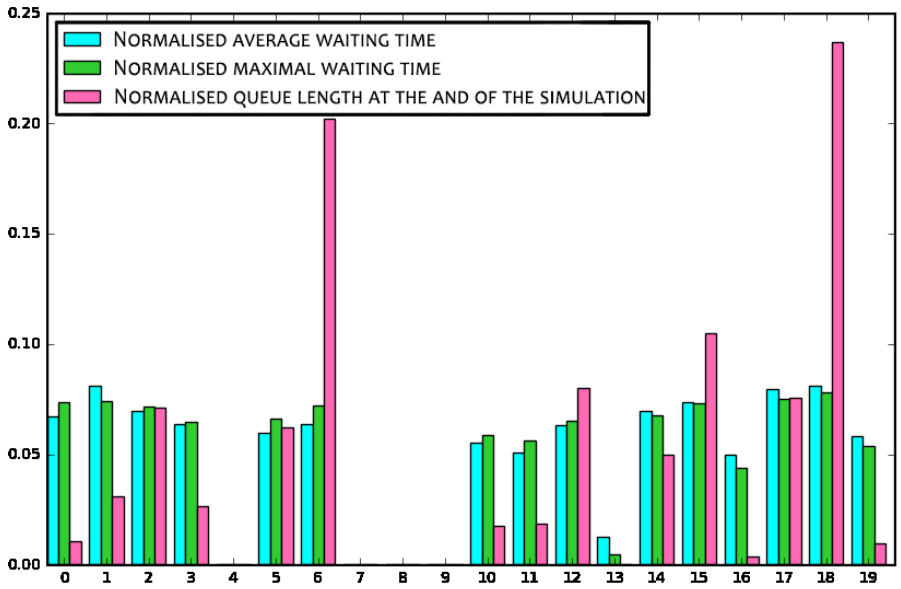


Figure 7-8 – Rush hour traffic: Queue length, maximal and average waiting times for each station.

Case 2: Off-peak demand

In Table 7.14 the main comparison results are shown for the seven strategies applied in the case where the demand is equal to 50% of the rush hour demand.

The obtained results are similar to those for the rush hour demand, where the SNN based strategies outperform the HNN based strategies, and the SNN+SDR improves slightly on SNN. SNN+IBR provides significant improvements over both strategies. In the off-peak demand case the increase of the performances, in terms of client waiting times (average and maximal) and queue length is even more important, but at the cost of a large number of empty vehicle runs. This can be expected as the cost of empty vehicle runs was not incorporated in the objective function (we focused purely on the client benefit in this subsection).

Mixing SNN+IBR strategies: The results of different mixes of the SNN and IBR strategies using the off peak demand are presented in Figure 7-9 and Figure 7-10.

These results show that under less demand conditions the addition of the IBR strategy to SNN dramatically improves the average waiting time at the end of the simulation. The SNN by itself has only a very modest effect on the average waiting time. This dramatic improvement by adding IBR is due to the heuristic nature of the IBR strategy, causing the vehicle to move to the demand station in anticipation of client demand.

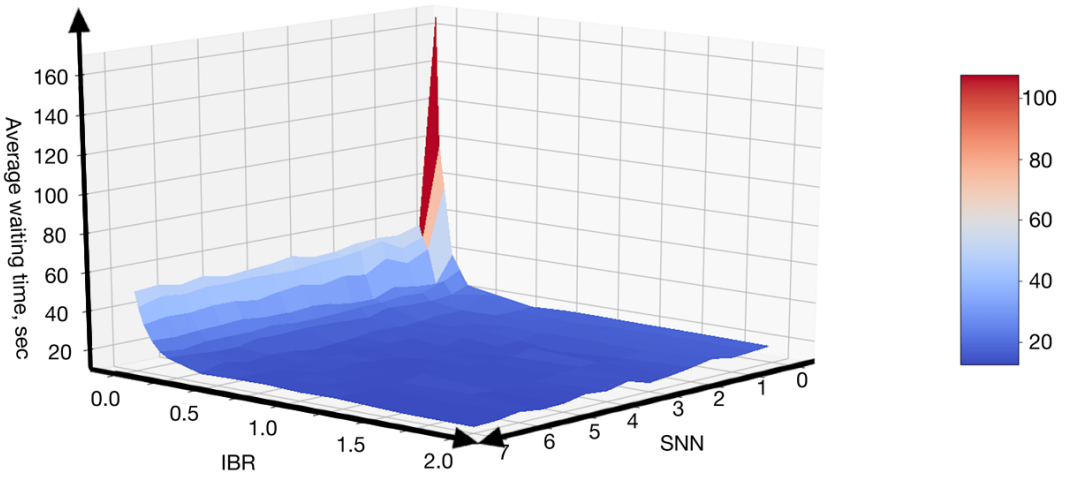


Figure 7-9 – Off-peak traffic: IBR-axis: Intensity of the IBR, times/second; SNN-axis: Intensity of the SNN, times/second; Z-axis: Average waiting time, seconds.

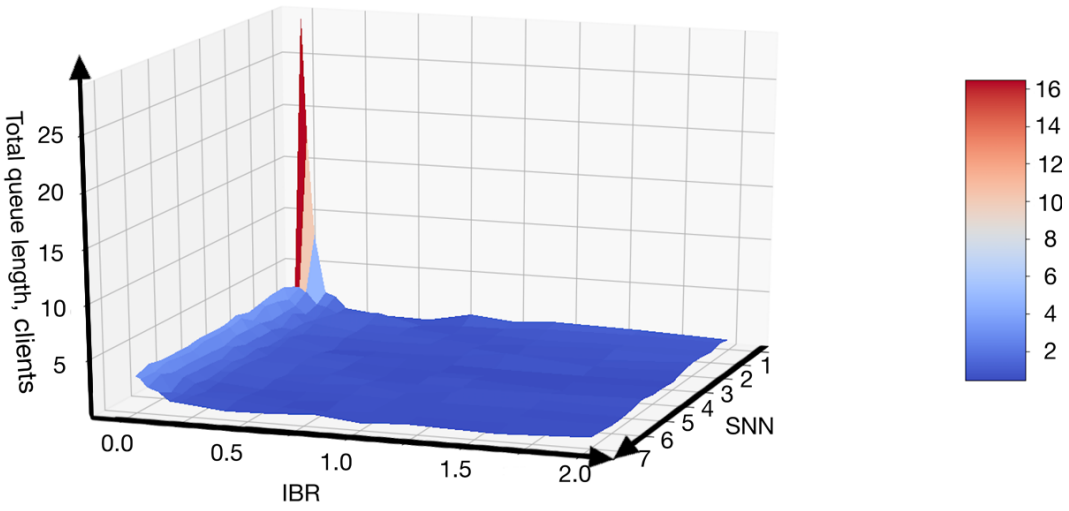


Figure 7-10 – Off-peak traffic: IBR-axis: Intensity of the IBR, times/second; SNN-axis: Intensity of the SNN, times/second; Z-axis: Average queue at the end of simulation, clients.

The main conclusion of these investigations is that the SNN+IBR mixed strategy provides the best results within considered strategies for both peak hour and off-peak demand cases, with the lowest maximum and average client waiting times and client queue lengths. However, total vehicle running time goes up as well, which is partly explained by the larger number of clients that are transported, but in the less demand case, the empty running contributes for a large proportion in the increase of the running time. Note that the algorithm formulations focus purely on minimizing the client waiting times, and do not include a cost term for empty mileage.

7.5.2 The matching problem

As it was described in Subsection 4.4.3, the rematchings can have an important influence on the system performance. We present the evaluation of rematching strategies on the Saclay network.

Based on the results from the previous subsection, the mixed SNN+IBR strategy was proposed as a basis for multimatching method. In order to evaluate the algorithm, firstly we choose a fleet-size of 60 vehicles and simulation time of 10000 seconds (around 2.8 h).

We compare three algorithm versions. The first is one-step mixing which is a redistribution algorithm. According to it, each time a single decision on redistribution of a single vehicle is about to be made, we choose randomly (according to predefined probabilities) one of the redistribution strategies described above. The ranking under chosen algorithm is then used to determine the top-rated station, and then the closest available vehicle is assigned to this station. This algorithm is computationally expensive since it only assigns a single vehicle to a station at each step.

In the greedy mixing and greedy-Hungarian algorithms, in each step we choose one redistribution strategy and assign all available vehicles to stations. In greedy mixing we do step by step the assignment between the top-rated station and the nearest available vehicle to it. In greedy-Hungarian algorithm we minimise the total run distance in order to serve all the top-rated stations. These two algorithms are less computationally expensive than the one-step mixing as they assign all available vehicles at each algorithm step.

Table 7.15 – The comparison between different mixing strategies for the case of off-peak demand.

Matching strategy	One-step mixing	Greedy mixing	Greedy+Hungarian mixing
Max wait (min)	6.3	7.2	7.1
Avg wait (min)	0.51	0.94	0.59
Avg queue (clients)	0	5	6

The results obtained show that the Greedy+Hungarian matching strategy improves the average waiting time in comparison to Greedy mixing. However, in terms of waiting times and queue length the one-step mixing outperforms both the greedy mixing and the Greedy+Hungarian method, albeit at a computational cost of having to re-run the redistribution algorithm for each available vehicle.

It is necessary to mention that although the brute force solution (not considered here due to its NP-complexity) may give a global optimum solution at any step, the total simulation still could deviate from the optimum. For example, if we apply SNN strategy to every vehicle, we will not have any vehicle available to employ proactive IBR strategy. Thus, further we can fail to meet higher future demand at any station. The next steps to balance the best reactive redistribution strategies versus ones, which leave empty vehicles to be redistributed in the future, requires further research.

7.5.3 Impact of ride-sharing

As a basis for ride-sharing method, we choose the SOTW strategy with the maximal delay caused by ride-sharing equal to 10 percent.

The results of the simulation on the Saclay network for fleet-size of 60 vehicles under morning rush hour demand with and without empty vehicle redistribution are shown in Figure 7.16. The simulation was performed using VIPSIM program and the simulation time was equal to 1 hour in order to evaluate system performance statistics on less simulation time.

As an empty vehicle redistribution algorithm, the one-step HNN method was used in this evaluation. The low simulation time comparatively to previous simulation changes the queue length, and the average and the maximal client waiting times. For example, in the case without empty vehicle redistribution, the waiting times differ less from the one in the case with empty vehicle redistribution, because there is not enough time to accumulate a sufficiently large queue to have a serious impact

Table 7.16 – The ride-sharing evaluation end EVR evaluation on Saclay under morning rush hour demand

	NRSH BA	SOTW SNN+IBR
Served (clients)	221	271
Not served (clients)	89	39
Max wait (min)	21.0	12.9
Avg wait (min)	3.6	2.5
Occ run (km)	1116.0	1044.1
Empty run (km)	0	313.8
Avg load (clients)	1.0	0.8
Avg queue (clients)	2.2	1.7

Table 7.17 – Rush hour demand: The impact of ride-sharing and mixed methods on average waiting time, in seconds.

Rush hour demand	40 vehicles in the system				
	BA NRSH	BA SOTW	EVR+SOTW		
			One-step mixing	Greedy mixing	Greedy-Hungarian mixing
	416	348	285	269	266
	60 vehicles in the system				
	BA NRSH	BA SOTW	EVR+SOTW		
			One-step mixing	Greedy mixing	Greedy-Hungarian mixing
	284	229	171	140	136
No-rush demand	40 vehicles in the system				
	BA NRSH	BA SOTW	EVR+SOTW		
			One-step mixing	Greedy mixing	Greedy-Hungarian mixing
	251	231	146	109	106
	60 vehicles in the system				
	BA NRSH	BA SOTW	EVR+SOTW		
			One-step mixing	Greedy mixing	Greedy-Hungarian mixing
	142	132	92	74	72

on these parameters. However, the simultaneous implementation of the ride-sharing and the empty vehicle redistribution has a visible impact on the number of served clients, the queue length and the client waiting times. Another effect is the more uniform structure of the average waiting time per station distribution.

The next simulations were performed on the Saclay network during 30 minutes, which is the short-term simulation. The evaluations are provided on a fleet-size equal to 60 and 40 vehicles for rush hour and off-peak demands, respectively.

The results presented in Table 7.17 show that both empty vehicle redistribution and ride-sharing have positive impact on the system performances. However, in contrast with results described above, the one-step mixing provides worse results than other mixed strategies. This can be explained by the fact that, when simulating during a long time period, it is important to balance out proactive and reactive redistribution strategies. When applying the simulation during a short period of time, all the vehicles can be redistributed on-line, even if continuing the simulation can lead us to an increase of the queue length.

The other parameters remain the same as in previous research (see Subsection 7.5.2). The investigation of the effect of the fleet-size and the demand variation on the system performances is the issue of the next subsection.

7.5.4 Impact of vehicle fleet-size

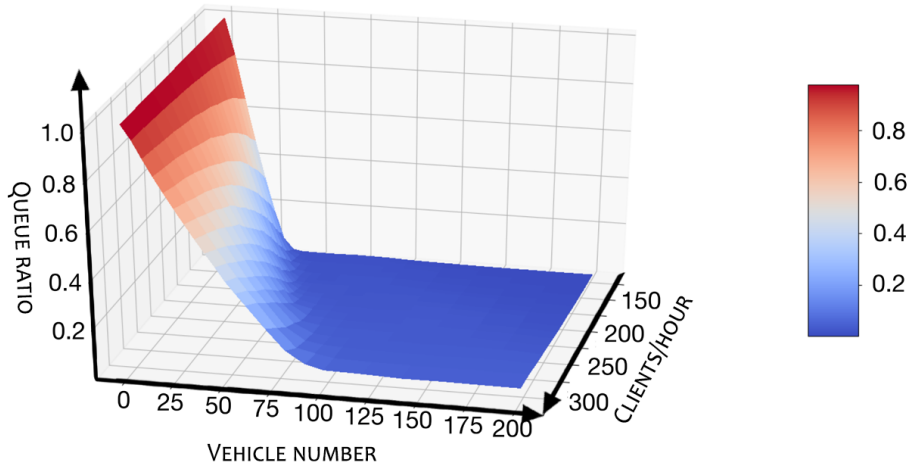


Figure 7-11 – Ratio of the queue length to the total number of the clients.

The results in Figure 7-11 were obtained using the mixed SNN+IBR redistribution (6 times per second for SNN strategy and 1.2 times per second for IBR strategy). In this figure we vary the client demand and the vehicle fleet size. The results show that increasing the number of vehicles decreases (as expected) the total queue length, for any level of client demand. The threshold level at which increasing the number of vehicles does not improve the queue length anymore varies linearly with the client demand, indicating that the algorithm is robust with regard to the demand level and the fleet size. The smoothness of the graph shows the robustness of this dependence, that is, its insensitivity to various deviations, related to the probabilistic nature of the mathematical models of clients' arrival and vehicles' redistribution.

7.5.5 Generalised costs

In the previous subsections we focused solely on the client waiting times and queue length at the end of the simulation for the objective function, leading to very good results in terms of the maximum and the average client waiting times, but at the cost of a large number of empty runs. In the following we investigate the effects of incorporating operating costs into the objective function, which is defined as follows:

$$F = \sum t_{wait} \cdot c_{wait} + \sum t_{run} \cdot c_{run} + N_{vehicle} \cdot c_{vehicle}.$$

The cost of client waiting time c_{wait} may depend on many factors, for simplicity we assume 10 euros/h. The cost of empty runs c_{run} depends on the cost of the electricity which we assume to be 0.45 euro/h of running time. Remember that in this scenario we consider speeds at edges to be constant (30 km/h). We assume that the cost of maintenance is around the same value, so $c_{run} = 0.9$ euros. The cost of deploying a vehicle $c_{vehicle}$, including the cost of insurance, was assumed to be 50 euros/day.

Figure 7-12 shows the results obtained for the case of off-peak demand with the total generalised cost as a function of the number of vehicles. The chosen optimisation strategy was the mixed SNN+IBR. The simulation time for this evaluation is equal to 10000 seconds. In this figure we see, that the total waiting time is the major component of the generalised cost, for vehicle fleet less than 40 vehicles. After that point, if we keep adding vehicles, the total waiting time keeps decreasing, but in terms of generalised cost it leads to the increase in vehicle deployment cost and the vehicle running cost. The optimal number of vehicles seems to be in the range of 40-60 vehicles, given the chosen values for c_{run}, c_{wait} and $c_{vehicle}$.

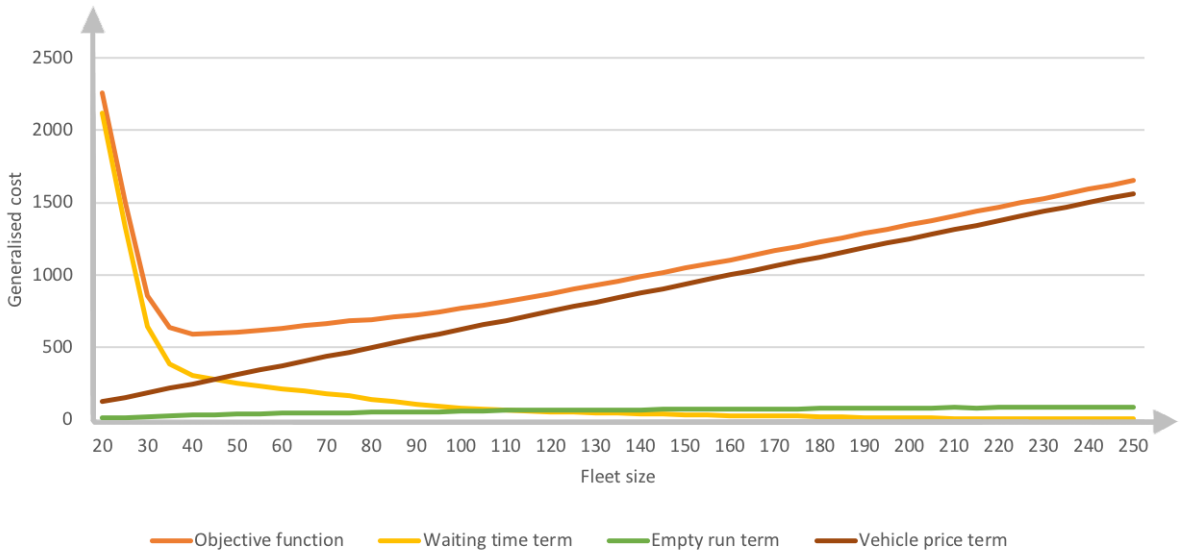


Figure 7-12 – Total generalised cost (in euro/day) as function of the number of vehicles under off-peak demand.

7.5.6 Summary

Seven variations of algorithm combinations were evaluated on a scenario without time windows of Paris Saclay network, France. The results show that the combination of Simple Nearest Neighbours + Index Based Redistribution (SNN+IBR) strategies provides promising results for both off-peak and peak-hour demand. However, the very good performances in terms of client waiting times and queue lengths comes at the cost of a large number of empty vehicle runs. This effect was especially noticeable for the off-peak demand.

Then we evaluate multimatching algorithms on the same network. While Greedy+Hungarian multimatching algorithm improves well known Hungarian method to get better results both in empty run time and in average waiting time, the one-step mixing shows the best results.

We also provide an analysis of the ride-sharing implementation and matching strategies with ride-sharing on the same scenario. The results show that both empty vehicle redistribution and ride-sharing have positive impact on the system performance. However, in contrast with results described in the matching subsection, the one-step mixing provides worse results than the other mixing strategies, because of shorter simulation period.

Finally, we introduce a generalised cost objective function that includes operator costs related to the number of vehicles deployed and the energy consumption of empty runs. The results show that there is an optimal number of the vehicles in the network, for both off-peak and rush-hour scenarii, where the client waiting times and the operator costs are balanced.

7.6 Evaluation scenario 4: Stockholm network

Evaluation of the algorithms for the empty vehicle redistribution problem with and without time windows is provided in this section using the scenario of Stockholm network. We evaluate the performance of the system under the described before combinations of algorithms under different conditions. In every case, the results are proposed as the solution of an optimisation subproblem on the algorithms intensities.

These algorithm combinations are compared under two conditions. First condition considers client waiting time window. We aim to study here the effects of clients refusing the service after waiting more than defined time threshold. Under second condition, the redistribution algorithm takes these time

windows into account. It is able to refuse to serve some clients that cannot be served by the system within these waiting time limits.

The first main objective is to study the effect of adding client waiting time windows to the simulation (the implementation of the time windows to the problem setting in comparison with the problem without time windows). The second main objective is to study the effect of adding time limited versions of the redistribution algorithms to take these time windows into account.

In the following we consider two different client arrival types:

- Stationary Poisson process,
- Real taxi data sets based arrival process.

7.6.1 Stationary Poisson Process based Simulation

We suppose that the clients arrive to the stations according to Poisson distribution with different constant rates for different stations. The time window size is 10 minutes, the number of vehicles is equal to 120 and they are distributed uniformly at the stations when the simulation starts. The simulations time is set to 10000 seconds.

The purpose of these experiments is to compare the system performances using the objective functions described in Section 3.5 (optimisation problem equations (3.5.1), (3.5.3), and (3.5.4)).

For every simulation case in this scenario we calculate the necessary number of replications. Formula (7.2.1) (see Section 7.2) is applied to all the considered objective values. Given the number of initial replications $m = 50$, $\alpha = 0.05$ and $\varepsilon = 0.1$, we obtain the required numbers of replications to obtain all the objective parameters, such as the number of served clients and their waiting times. For example, for the SNN+IBRTL combination, the formula shows that the required number of replications is equal to 84. The number of replications does not exceed 100 for any of the algorithms combination and for any of the objective parameter. Thus 100 simulations is used across all of the algorithm combinations.

The comparison of redistribution strategies for the problem with time windows

Table 7.18 provides the first performance measures computed. We investigate two cases here. First, clients with time window (left 4 columns group) where clients not served within 10 minutes will quit the system. Second, clients wait until they are served (last 3 columns group).

Table 7.18 – Redistribution strategies without TL for Stockholm network.

	with time windows				without time windows		
Combination	*Not served (clients)*	*Avg wait (min)*	*Max wait (min)*	*Avg queue (clients)*	*Avg queue (clients)*	*Avg wait (min)*	*Max wait (min)*
1.1 SNN+SDR	129.44	6.2	10	35.22	115.6	11.4	27.7
1.2 SNN+IBR	118.36	6.2	10	31.76	48.9	9.1	26.5
2.1 HNN+SDR	238	5.8	9.99	30.78	111.6	11.5	28.9
2.2 HNN+IBR	224.98	5.8	9.99	23.76	124.0	13.4	33.4
4.1 STN+SDR	59.82	2.4	9.91	11.84	16.9	3.1	27.5
4.2 STN+IBR	**55.5**	**2.9**	**9.95**	**9.58**	**9.1**	**3.455**	**18.2**

Not surprisingly the results in Table 7.18 show, that in terms of the average waiting time, the algorithms applied to the case with time window provide better solutions. Nevertheless, this difference is obtained by the bigger number of not served clients. In different problems occurring in transportation, the utility function of clients contains not only the waiting time, but also if the waiting was successful. Thus, when choosing between strategies with close average waiting times, the number of served during the simulation clients and the queue length have to be taken into account. The combination with STN algorithm can in this case help to improve the number of served clients and the average waiting times within served clients.

Effects of waiting time windows

In the case where client time windows are considered, the application of the algorithms with abbreviation TL (time limited) introduced in Subsection 4.4.2 is possible.

Table 7.19 – Redistribution strategies: the problem with time windows.

Combination	*Not served (clients)*	*Avg wait (min)*	*Max wait (min)*
1.3 SNN+IBRTL	80.14	5.9	9.99
2.3 HNN+IBRTL	112.24	6.2	9.99
3.1 HNNTL+SDR	110.12	6.2	9.99
3.2 HNNTL+IBR	85.92	6.1	9.99
3.3 HNNTL+IBRTL	97.98	6.0	9.98
4.3 STN+IBRTL	**37.42**	**2.7**	**9.96**

Comparing Table 7.18 and Table 7.19 reveals that the combinations with time limited variations of algorithms outperform combinations without TL, in terms of the number of clients served within the time windows. Nevertheless, in terms of average waiting times of the clients served, there are no significant differences. The maximal waiting times, obtained for all the algorithms considered for the problem with time windows, are close to the time window sizes.

The results also show, that in combination it seems to be sufficient to use only one of the heuristical algorithms which takes into account the clients' time windows. For example, when the reactive strategy is HNNTL, the change of proactive one from IBR to IBRTL does not improve the overall performances. The most robust combination for the case with time windows in redistribution algorithm is STN+IBRTL and, in comparison to STN+IBT, considering time limits in redistribution helps to decrease most attributes. The most significant drop is in the number of not served clients from 55.5 to 37.42 which is about 32 %.

Effects of the demand

To show the robustness of the empty vehicle redistribution strategies proposed, it is important to evaluate this performances at different demand levels. In the scenario of high demand level, serving all the clients is not possible. Off-peak hours demand demand can theoretically be served completely or almost completely. The results of the same algorithms applied to this case (by considering 50% of the original demand) are shown in Table 7.20.

Table 7.20 is organised the same way as Table 7.18. The «—» in the table means that these data are not available, because TL versions of algorithms cannot be used for that case. The simulation results show, that for the case with time windows, the combinations SNN+IBRTL, HNNTL+IBR, and STN+IBRTL seem to be promising in terms of the number of clients served within the time window. For the problem without time windows, the combination STN+IBR shows the best results in terms of both average and maximal waiting times. Here, the difference between STN based algorithms and SNN ones is not so significant for both cases, with time windows and without time windows.

The maximal client waiting times in the case with time windows show, in contrast to full demand scenario, smaller values than the size of time windows.

The difference in the results of the simulations with and without clients' time windows is not significant for identical methods. For example, when applying the STN+IBR strategy, the average maximal waiting time within served clients is 6.43 minutes for the case with time windows and 6.01 minutes for the case without time windows; while the average waiting times are 0.40 and 0.45 minutes, respectively. However, the total number of not served passengers is 0.06+0.18=0.24 (the number of clients that have quitted the system plus the number of clients remaining at the end of the simulation) in the case with time windows and 0.2 in the case without time windows. Thus, the overall performances are of the same order.

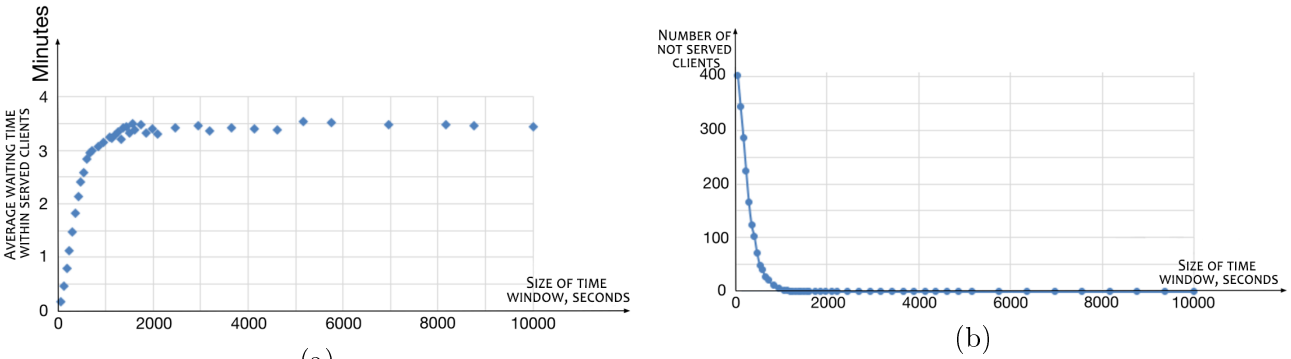
Table 7.20 – Redistribution strategies for Stockholm network, small demand.

	with time windows				without time windows		
Combination	*Not served (clients)*	*Avg wait (min)*	*Max wait (min)*	*Avg queue (clients)*	*Avg queue (clients)*	*Avg wait (min)*	*Max wait (min)*
1.1 SNN+SDR	1.72	1.33	7.67	1.94	1.24	1.14	7.85
1.2 SNN+IBR	0.02	0.68	6.32	0.02	0.14	0.67	6.18
1.3 SNN+IBRTL	0.14	0.67	6.22	0.12	—	—	—
2.1 HNN+SDR	2.14	1.31	7.87	1.98	1.18	1.21	7.96
2.2 HNN+IBR	0.22	0.65	6.18	0.2	0.16	0.76	6.64
2.3 HNN+IBRTL	0.14	0.71	6.23	0.18	—	—	—
3.1 HNNTL+SDR	1.66	1.15	7.43	2.2	—	—	—
3.2 HNNTL+IBR	0.06	0.66	6.04	0.12	—	—	—
3.3 HNNTL+IBRTL	0.16	0.79	6.75	0.06	—	—	—
4.1 STN+SDR	0.46	0.66	7.52	2.04	0.62	0.73	8.42
4.2 STN+IBR	0.06	0.40	6.43	0.18	0.2	0.45	6.01
4.3 STN+IBRTL	0.02	0.43	6.02	0.12	—	—	—

For the case with time windows, the STN+IBRTL strategy shows the best results for all performance measures.

Influence of the time window size on the system performances

To provide a reasoning for the stated level of service in autonomous taxi systems, the analysis of the time window size is provided. Considering that the time window is equal to the time limit in TL versions of the algorithms combinations, we provide the performance analysis for the best evaluations in terms of clients served for the full demand scenario, STN+IBRTL. The aim is to show the average number of clients served, and the average and maximal waiting times depending on time window size.



(a) (b)

Figure 7-13 – (a) Time window impact on the average waiting time for served clients. (b) Time window impact on the number of not served clients.

The results of the simulation show a shape with negative gradient in terms of the number of clients not served within their time window constraints (Figure 7-13 (a)). In terms of the average waiting times for served clients, Figure 7-13 (b) shows that until reaching the critical time window size values, the average waiting times for served clients will be oscillating around 3.5 minutes (whenever the maximal waiting times can be significantly big and achieve the current time window size).

7.6.2 Simulation on real demand data

The third simulation uses real demand data, using the taxi departure time of every trip provided in the Stockholm taxi data set. These data are described in Subsection 6.5.4. Nevertheless, the IBR and

IBRTL algorithms are assuming the Poisson nature of predicted future arrivals. In this simulation, the comparison between different redistribution algorithms in Table 7.21 shows, that in terms of the number of not served clients, the combination STN+IBR outperforms the other combined algorithms. For the problem without time window constraint, this combination significantly outperforms the others in terms of the maximal waiting time. In the problem without time windows, the best performance in terms of queue length was obtained by the combinations with STN, and especially STN+SDR strategy.

Table 7.21 – Results for Stockholm network with real demand data.

	with time windows				without time windows		
Combination	*Not served (clients)*	*Avg wait (min)*	*Max wait (min)*	*Avg queue (clients)*	*Avg queue (clients)*	*Avg wait (min)*	*Max wait (min)*
1.1 SNN+SDR	118	5.47	10	143	249	6.03	25.1
1.2 SNN+IBR	106	5.27	10	125	284	7.99	36.66
1.3 SNN+IBRTL	81	4.92	9.97	122	—	—	—
2.1 HNN+SDR	240	2.95	9.99	132	358	7.09	30.04
2.2 HNN+IBR	263	3.74	10	137	286	5.35	33.84
2.3 HNN+IBRTL	56	3.84	10	116	—	—	—
3.1 HNNTL+SDR	83	3.49	9.96	125	—	—	—
3.2 HNNTL+IBR	81	4.33	9.99	124	—	—	—
3.3 HNNTL+IBRTL	76	3.91	9.98	123	—	—	—
4.1 STN+SDR	48	1.91	9.95	57	68	2.43	31.83
4.2 STN+IBR	39	1.98	9.93	75	86	2.97	21.40
4.3 STN+IBRTL	43	2.62	9.97	84	—	—	—

The strategies HNN+SDR and HNN+IBR show the worst performances in terms of the number of not served clients for the case with time windows. Nevertheless, the HNN+IBRTL combination shows better results than SNN+IBRTL. This can be explained by a reference to the formulation of the algorithms: in HNN algorithm, the vehicle is to be sent to the (at the moment of the nearest vehicle arrival) waiting client with maximal time, without taking into account the fact that the client could have already quit the system because it had to wait more than its time window.

7.6.3 Summary

Using computational experiments on the Stockholm network scenario under generated Poisson and real taxi data sets for the demand, we investigated the effects of two main study conditions. The first condition is the time window concept that adds the constraints to the client waiting times. The second condition tested is the concept of incorporating the client waiting time limit in the redistribution algorithms, thus the system can reject demands that will be not served. Considering client waiting time windows allows modelling them in more realistic way because in the reality no one is willing to wait infinite time for service. By second condition we involved this time limit consideration to the redistribution algorithms which helps to improve almost all performance measures we studied. Under this condition, new IBRTL and HNNTL algorithms that improve on existing empty vehicle redistribution algorithms are evaluated. The number of not served clients can be lowered by about 32% due to adaptive redistribution towards waiting clients.

The results presented in this section show that the IBR based strategies perform well for provided demand, especially in combination with existing nearest neighbours strategies such as SNN and STN. The results on Stockholm network show, that STN+IBRTL redistribution strategy is one of the most robust and best performing for the problem with time windows. The implementation of proposed algorithms to the real demand case shows that STN based strategies of empty vehicle redistribution show the best results in terms of the number of clients served within their constraints, and STN+IBR strategy shows the significant improvement of maximal waiting time for the problem without time windows.

7.7 Evaluation scenario 5. Porto network

In this section, we investigate the performances of the learning process of the zone-based RL method using the training and test data sets of the city of Porto (Subsection 6.5). Firstly, we provide the analysis of city road graph partitioning. Then we evaluate the system performance in cases of clients with deterministic and stochastic preferences. We study all the main system quality characteristics, such as the number of satisfied clients and their waiting times, in function of the demand level in the system and the fleet-size.

We start the reinforcement with learning on training data set of 364 days with 1310196 demands. The results are to be compared on test data set of 1 day with 3725 demands. Remind, that in real network the number of vehicles is equal to 442 (Subsection 6.5). The comparison between vehicle battery characteristics and their charging strategies is not a topic of interest in this section. We use predefined strategies, charging and parking stations, and their characteristics, as described in Section 6, identical for all the simulations below. We considered also parking stations distributed over the city road network graph using methods of k-medoids. The resulting distribution provides the average distance between adjacent stations which is equal to 1.24 kilometres. For parking stations with charging slots this average distance is 2.1 kilometres.

7.7.1 Zone decomposition

The preliminary part of the proposed method consists of zone repartition. Remind, that the number of vertices in the network is equal to 3705, whereas the number of arcs is 8913. Figure 7-14 provides information about obtained zone diameters depending on the number of zones. These data show, that the average diameter of the zone inversely corresponds to the number of clusters and is of smooth nature. Nevertheless, the maximal diameter has more awkward nature but it remains decreasing. The biggest zone is of 2.177 kilometres diameter and corresponds to a specific zone, bounded by Douro river. The zone decomposition, as will be shown in further subsection, influences both the system performances and the processor execution time. The search of equilibrium between these values is one of the main issues of this work.

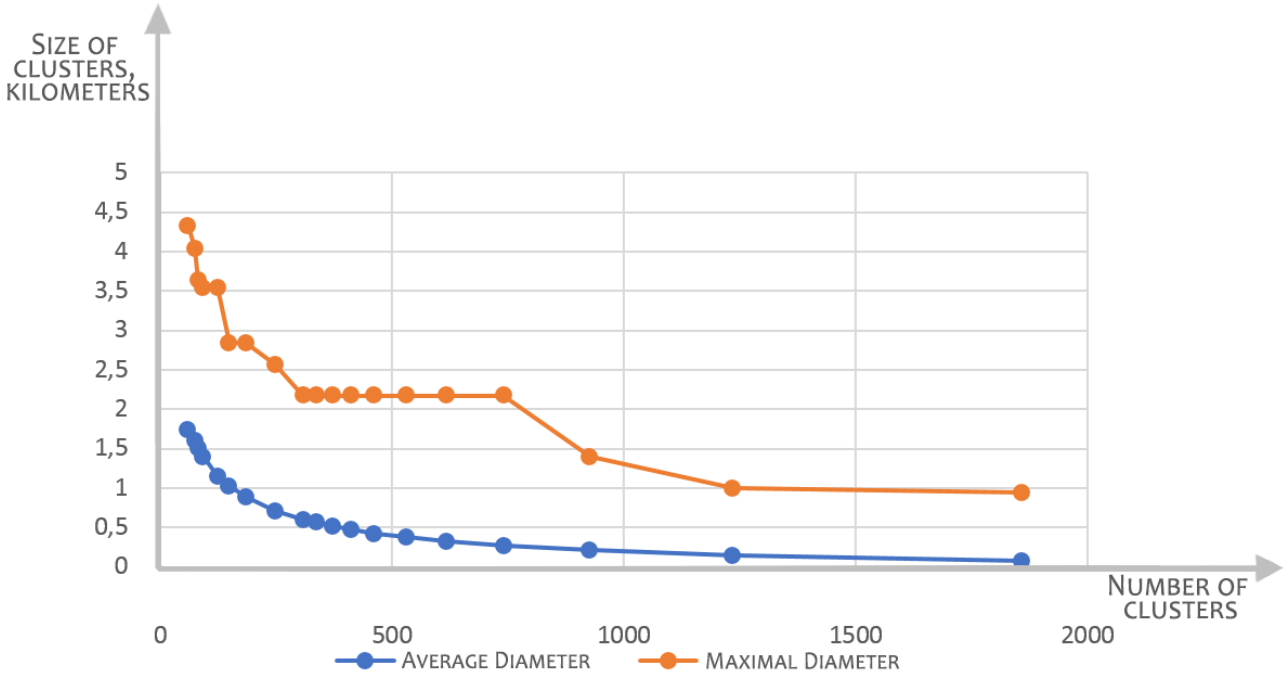


Figure 7-14 – Zone sizes obtained by FCM.

7.7.2 Deterministic clients preferences

We consider clients with deterministic preferences. The time window size is fixed to 10 minutes and the walking distance to 100 metres. All the clients agree to share their ride.

The fleet-size evaluation

We compare the results for different zone decompositions on the fixed above evaluation objectives. Figure 7-15 provides the dependency between the number of served clients for purely heuristical methods without reinforcement learning and zone decomposition from the fleet-size. The results show, that dependency of number of served clients from available vehicles is monotonic non-descending. The results obtained on cases with and without ride-sharing differ in their nature. In the case with ride-sharing, the number of satisfied clients grows up fast in the beginning and approaches the maximal value of around 50 vehicles in the system. However, in the case without ride-sharing, after the fast increase until 30 vehicles in the system, the number of served clients stagnates around 3000, and remains of the same order until we reach 70 vehicles in the system. The number of served clients differs the most for cases with and without ride-sharing for fleet-size between 35 and 75 vehicles. This can be explained by the fact, that under small fleet-size, the average load is smaller because of the lack of sharing possibilities.

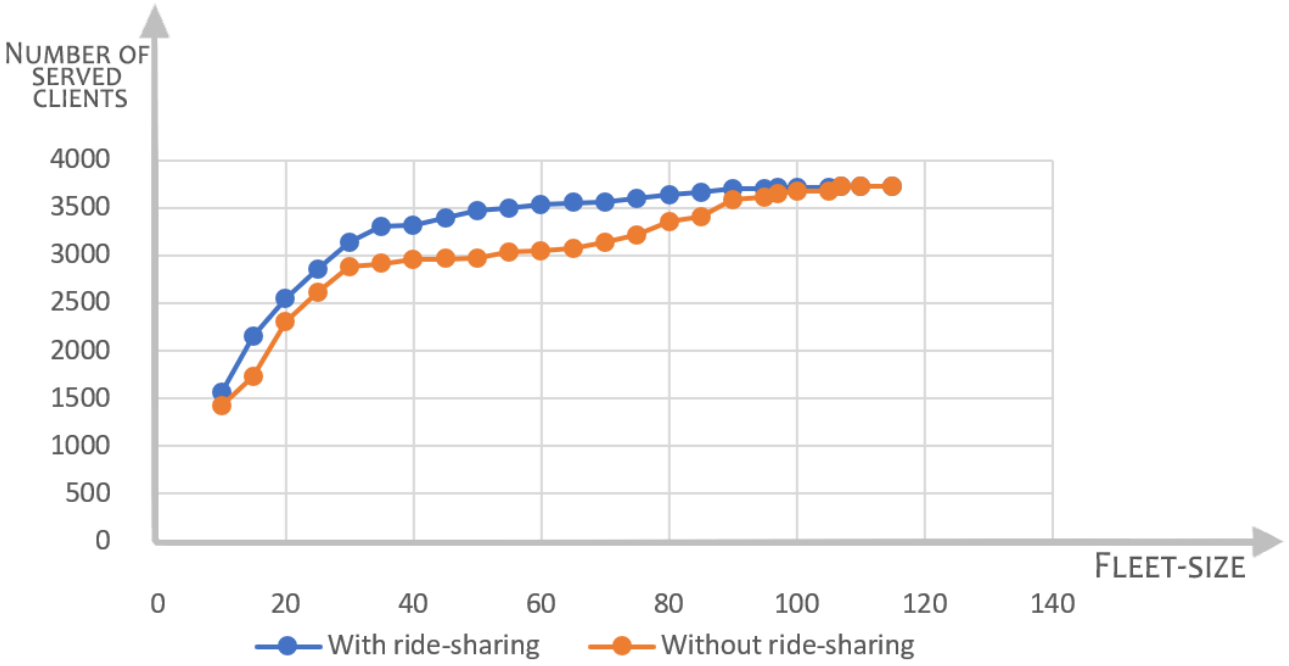


Figure 7-15 – The dependency between number of served clients and the fleet-size.

Afterwards, we evaluate 4 possible cases: with and without accepted ride-sharing together with enabled or disabled zone-based RL. Table 7.22 consists of the minimal required number of vehicles to serve 3600 clients. Based on the previous results, we assume that dependency between served clients and available vehicles is monotonic non-decreasing and we use binary search to find the threshold values. The reinforcement learning stage is processed on the training data set of Porto, considering the number of zones $|Z|$ to be equal to 123. The obtained results show that when the RL implementation uses additional execution time, the fleet-size is used more effectively.

Table 7.23 provides the system performance evaluation using 80 vehicles and 2 different values of the number of clusters, 123 and 336. The results show, that with 336 clusters, a bigger number of served clients is obtained with increased waiting times and execution time. The small average load in the system evaluation can be explained by a non intensive passenger flow. However, similar loads

Table 7.22 – The required fleet size with and without ride-sharing and reinforcement learning (|Z|=123.)

|Z|= 123	NRSH No RL	NRSH RL	SOTW No RL	SOTW RL
Number of required vehicles	97	95	75	75
Exec time (sec)	21.3	39.7	13.4	41.1

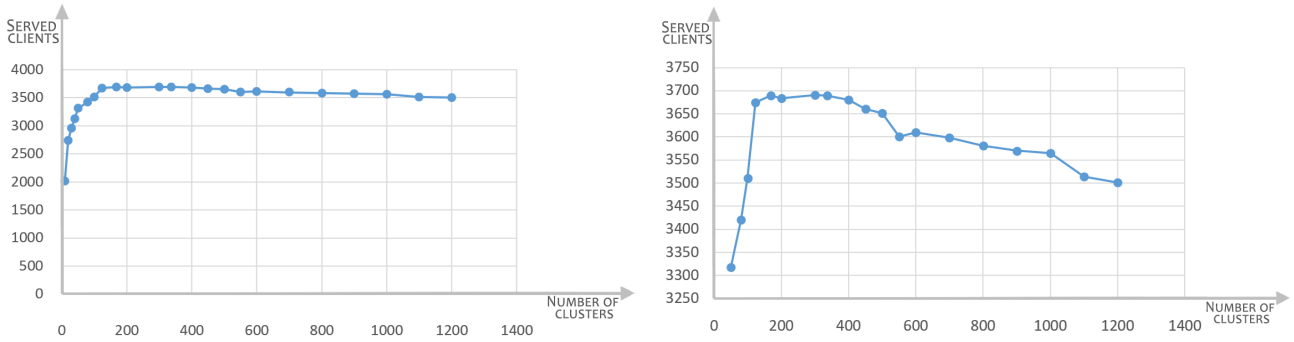
in cases with and without ride-sharing can be justified. There are more empty runs in the case with enabled ride-sharing and at the same moment there are more served clients.

Table 7.23 – Main results for the fixed fleet-size of 80 vehicles, |Z|=123 and 336.

	|Z|= 123				|Z|= 336			
	NRSH no RL	NRSH RL	SOTW no RL	SOTW RL	NRSH no RL	NRSH RL	SOTW no RL	SOTW RL
Avg wait (min)	3.56	3.69	2.67	2.76	3.31	3.51	2.85	2.97
Max wait (min)	9.99	9.96	9.98	9.98	9.97	9.98	9.99	9.97
Served (clients)	3521	3562	3638	3674	3531	3581	3572	3689
Avg load (clients)	0.51	0.53	0.54	0.54	0.50	0.52	0.52	0.53
Exec time (sec)	11.6	35.9	15.1	40.2	29.2	114.3	33.3	136.6

Influence of the number of zones on the system performances

This subsection provides evaluation of the system performances when varying the number of zones in the system. All the simulations are provided for 80 vehicles in the system. We study the system performances expressed in terms of the number of satisfied clients and execution time. The execution time results in Figure 7-16 show an increase with the increase of the number of zones. However, this increase remains close to linear one, thus, the non-centralised zone-based reinforcement learning with combinatorial optimisation methods can be provided in real time even in huge networks. The number of served clients has another behaviour. It increases, as it is shown in Figure 7-17a, until the boundary number of zones around 150. Till around 400 zones in the system the number of served clients is around 3689. Under more then 400 zones this number starts to slowly decrease, as it is shown in more detailed Figure 7-17b. The decreasing nature of this curve under big number of zones can be explained by the fact, that smaller granularity leads to less predicted nature in empty vehicle redistribution and slower learning because of not enough requests between zones during the training period to learn exact patterns. For later simulations, we chose the number of zones $|Z| = 168$. This number provides at the same moment the optimal amount of served clients and the minimal processor execution time.



(a) (b)

Figure 7-17 – The dependency of the number of served clients from the number of clusters in the system. (a) General data for |Z| from 10 to 1200. (b) Detailed results for |Z| of 50 and more.

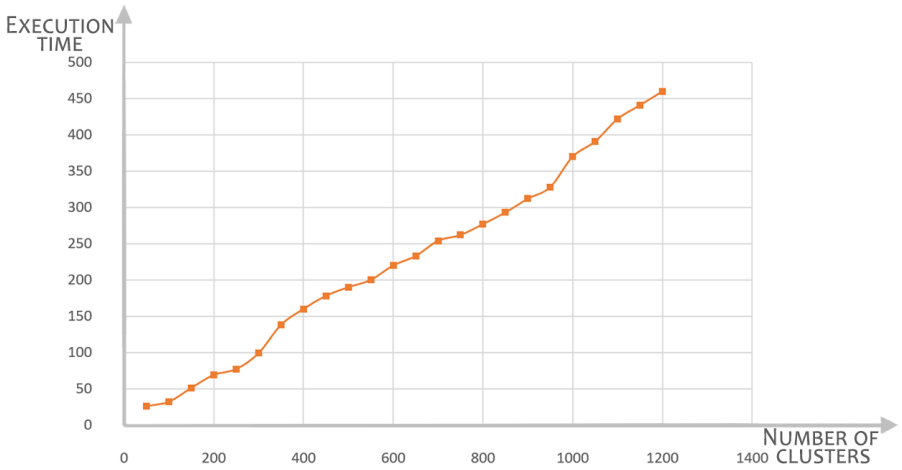


Figure 7-16 – The dependency of the execution time on the number of clusters in the system.

Start positions of the vehicles

This Subsection is an intermediate link between the dynamic modelling (online optimisation) and the case of the complete information (offline optimisation). Our objective is to choose the start positions of the vehicles in order to satisfy during the simulation the maximal number of clients. Based on previous results, we set the number of zones |Z| in the system to be equal 168, as a compromise between the number of served clients and the execution time.

In every discussed above simulation, the vehicles were distributed uniformly in the beginning of simulation on test data set. The results on uniform vehicle initial distribution reveal 3689 served clients within their constraints (Table 7.24).

The basic heuristic of the vehicle initial distribution assumes that if the first clients are supplied by vehicles, they will be served. The corresponding vehicle initial distribution with full ride-sharing in the system let us to satisfy 3695 clients.

Another heuristic uses the method of differential evolution. This method is used to maximize the objective function that depends on the argument vector. The genetic algorithm underlying this method uses techniques such as population crossing and mutation. At each step, the strongest individuals are selected to give the best values to the objective function (the number of satisfied clients). In this scenario, the initial population is consisted of 32 individuals, each is initially placed randomly (uniformly) before start of the simulation. After 50 selection steps, the individuals with the best value of the objective function were picked. The use of the DE method allows satisfying 3701 clients.

However, the better results in terms of the number of served clients are obtained at the cost of the average waiting times of served clients. This can be explained both by taking into account only served clients within their constraints in waiting time calculation and by other movements of the vehicles. The low rates of shared trips in all considered cases are explained by the intensity of the client arrivals and their distribution.

Fleet-size evaluation

We provide the evaluation results of the fleet-size for different learning levels. Here, we chose methods of zone-based RL, centralised zone-based RL and "look with few steps forward" heuristic under complete information (Subsection 5.6.2). For the evaluation of the system performances, expressed in terms of the number of served clients, in the case of the complete information we use the heuristic with 2 steps in advance.

Figure 7-18 shows the dependency of the number of served clients from the fleet-size under mentioned above learning levels. The results are provided in all cases with fully shared vehicles. The number of zones is chosen as compromise between the execution time and the number of served clients,

Table 7.24 – Comparison between start vehicle positions.

	Uniform	First	DE
Avg wait (min)	3.03	3.15	3.17
Max wait (min)	9.83	9.89	9.55
Served (clients)	3689	3695	3701
Empty run (km)	10110.7	10267.3	10120.0
TR_1 *(km)*	14158.7	14313.7	14268.1
TR_2 *(km)*	276.8	243.9	233.4
TR_3 *(km)*	13.3	7.5	4.6
TR_4 *(km)*	0	2.6	0

and is equal to 168.

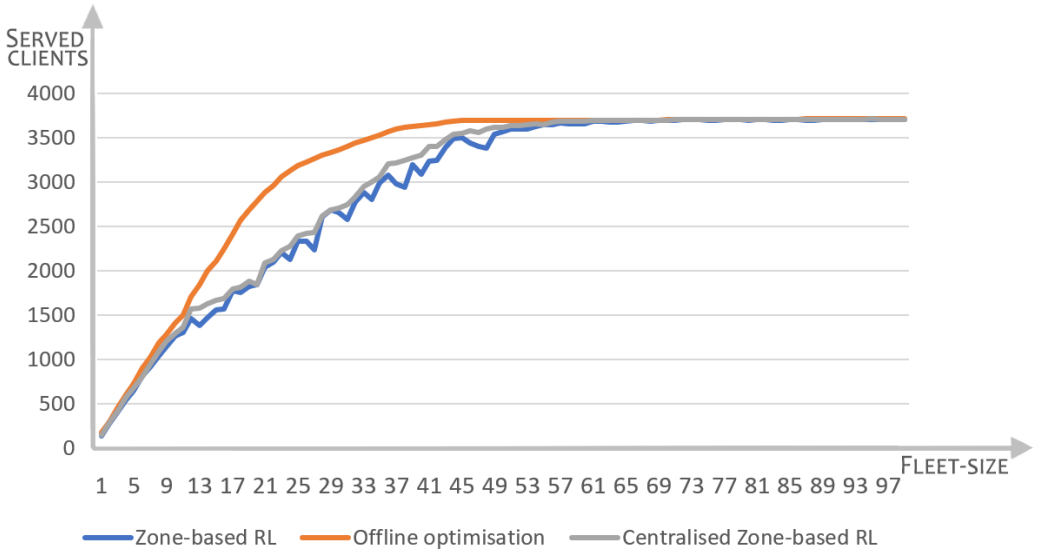


Figure 7-18 – The fleet-size influence on the number of served clients.

In each test simulation, vehicles are initially distributed uniformly over the network. However, this distribution affects the further vehicle movements. As a consequence, the number of served clients under zone-based reinforcement learning does not show strictly monotonous behaviour, especially under small number of vehicles. Nevertheless, the centralisation, which leads to re-matchings, allows obtaining a nearly monotonous behaviour of the number of served clients. The heuristics for offline problem also provide monotonous dependency between fleet-size and the number of served clients.

Figure 7-18 shows, that the offline optimisation outperforms both zone-based RL and centralised zone-based RL for number of vehicles between 5 and 57. After, all three methods are able to serve nearly all the clients. However, the methods for offline optimisation can be used only as a benchmark for other methods because of a-posteriori information requirement.

Demand/fleet-size evaluation

We study the influence of the demand and the fleet-size on the system performances. We consider fixed distribution of the clients, but we change their arrival rates, proportionally, and the variation is the time period where all clients arrive. Figure 7-19 shows the results of the relative simulations under the following conditions. The X-axis corresponds to the client arrival rate (clients/hour). The used optimisation is zone-based reinforcement learning method with $|Z|$=168 which was shown promising. The non-proportional growth related to the ratio of the demand and the fleet-size corresponds to possible increase in the ride-sharing possibilities under more dense demand.

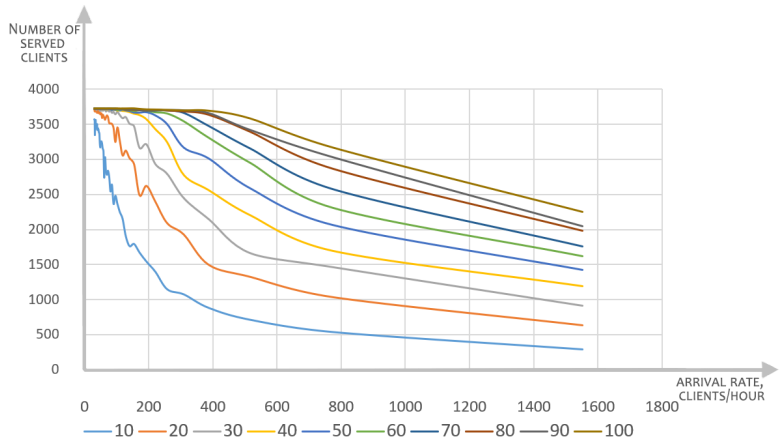


Figure 7-19 – Influence of demand and fleet-size on the number of served clients. Lines of different colours correspond to the number of vehicles in the system. X-axis: Arrival rate (clients/hour). Y-axis: Number of served clients.

Based on statistics under variation of fleet-size from 10 to 100 vehicles and demand in the system with arrival time proportionality between 0.1 to 5, as described in Figure 7-19, we obtain the scatter diagram. The function of the number of clients served from the ratio of the fleet size to the number of arrivals per hour has a non-decreasing value. However, starting from a certain size of this fleet the number of clients served ceases to increase as a saturation state occurs. From this point on, it becomes inefficient to increase the number of vehicles. Figure 7-20 shows the value of this function. This scatter plot is obtained by change in the size of the fleet from 10 to 100 vehicles and demand in the system with the proportionality of arrival time from 0.1 to 5. The boarder value of around 0.2 corresponds to the rate, when clients in the system are served nearly to their totality.

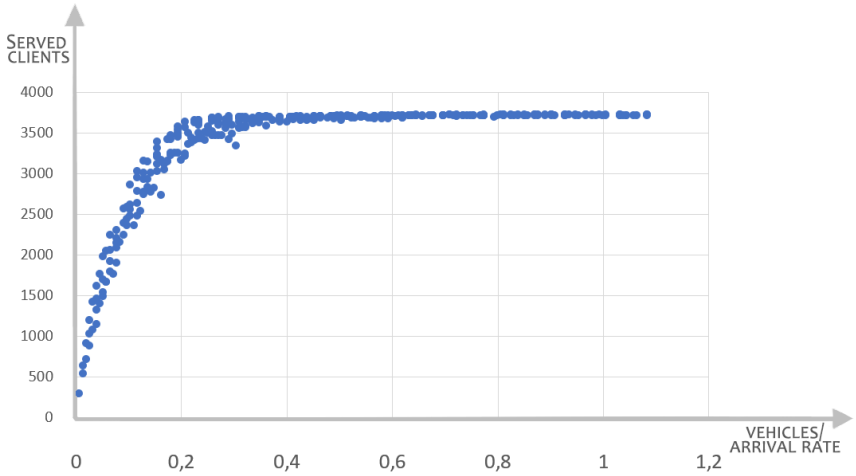


Figure 7-20 – Influence of the demand and the fleet-size on the number of served clients.

7.7.3 Stochastic clients preferences

In the following we study the system performances in the case of stochastic clients. In this case, client generation is based on clients of different types, as described in Section 6.3. Remind, that these types of clients differ on their utility function, call time, ride-sharing preferences, as well as on their acceptable walking distance and waiting times. We consider 4 different client types.

Client preference repartition

Let us fix the fleet size to 80 vehicles, which, based on Subsection 7.7.2, is a boundary value for served clients with deterministic preferences. We fix the number of zones to 168. The types of clients are those described in Section 6.3. Clients of different types are distributed uniformly. The ride-sharing preferences for clients of types 2 and 4 are distributed uniformly. Thus, 50% of clients of each of these types have low acceptability and 50% have bigger one.

Table 7.25 – The system performances under different client distributions.

Characteristics	Set 1	Set 2	Set 3	Set 4	Set 5
Type 1, %	25	100	0	0	0
Type 2, %	25	0	100	0	0
Type 3, %	25	0	0	100	0
Type 4, %	25	0	0	0	100
Avg queue (clients)	3.56	4.43	3.82	3.84	1.49
Max wait (min)	28.10	29.11	10.00	10.00	3.00
Served (clients)	3082	3718	3461	3457	2424

Table 7.25 shows the effects of client preferences repartition on the system performances. We see, that the results of Set 2, which consists totally of clients of type 1, who are the most tolerant to waiting times, show the best performances. Set 5, which consists of clients of type 4, who call vehicles in advance, but are not ready to wait, shows the worst performances. This can be explained by the fact, that zones and vehicles provide the priority to nearest client arrivals, and prefer to not arrive to specific stations in advance, in order to diminish useless waiting times. Set 3 and Set 4 show similar results. In these sets, the clients characteristics are similar, and the main difference is the utility functions of the clients.

Fleet-size evaluation under demand variation

We provide the evaluation of the client arrival rates and fleet-size on the number of satisfied clients. The used optimisation is zone-based reinforcement learning method with 168 zones and the client preferences correspond to those of Set 1. In the same logic, as for deterministic clients preferences, we obtain this dependency on of number of served clients and time proportionality of the demand.

Figure 7-21 shows the dependency between the ratio of vehicles in the system to client arrivals per hour and number of satisfied clients. These results were obtained as follows. Based on statistics under variation of fleet-size from 10 to 100 vehicles and demand in the system with arrival time proportionality between 0.1 to 5, by the same methods than in previous subsection, we obtain the scatter diagram. Similarly to previous results, the scattered results remain increasing, but with bigger standard deviation. The dispersion of the obtained results can be explained by stochastic nature of client preferences, which are not known by vehicles in advance. Thus, non-satisfaction of some clients can happen even with big fleet-size. The boarder value of around 1 corresponds to the rate, when clients in the system are served nearly in their totality. However, starting from ratio of around 0.2, the increasing number of served clients is slowing down, then the expectancy of this number slowly approaches to its possible maximum.

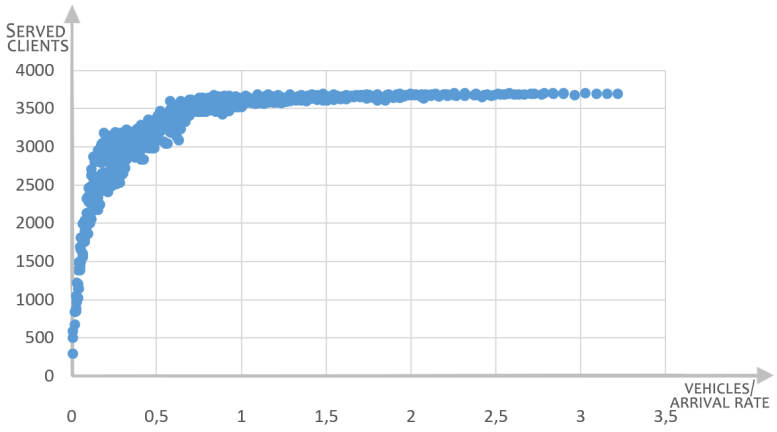


Figure 7-21 – Influence of demand and fleet-size on the number of served clients.

Different learning levels evaluation

Similarly to the case of deterministic client preferences, we evaluate the system performances for the case where clients are distributed as in Set 1. We study the system using methods of zone-based RL, centralised zone-based RL and "look with few steps forward" heuristic under complete information (with 2 steps). Figure 7-22 shows the dependency of the number of served clients on the fleet-size for these three learning levels.

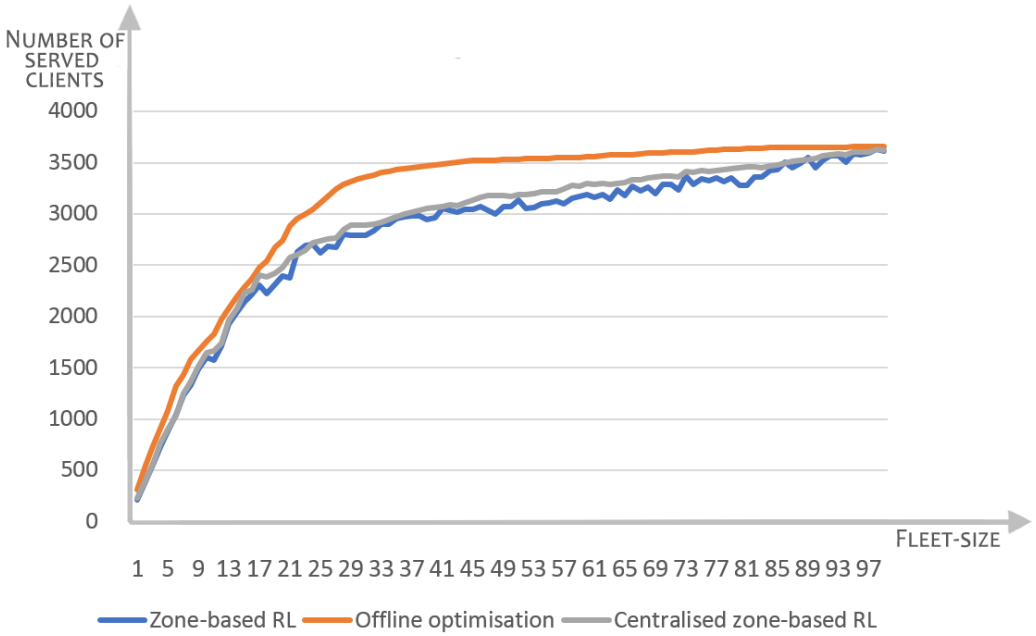


Figure 7-22 – The fleet-size influence on the number of served stochastic clients.

The number of served clients under zone-based reinforcement learning does not show strictly monotonous behaviour, especially with small number of vehicles. The centralisation, which leads to re-matchings, allows to obtain more monotonous nature of corresponding curve. However, there are small fluctuations, in contrast to the case of deterministic client preferences. The heuristics for offline problem provide monotonous dependency between the fleet-size and the number of served clients. The figure shows, that the offline optimisation outperforms for this client distribution both zone-based RL and centralised zone-based RL.

7.7.4 Summary

In this section we studied system performances using different methods for aTaxi system management.

The focus was made on evaluation of methods developed for huge networks, which is based on problem decomposition and scalability. Problem decomposition includes the application of combinatorial optimisation methods for a set of specific subproblems, such as charging management, empty vehicle redistribution and ride-sharing. The scalability is achieved by urban zone partitioning and larger learning agents. The presented results show that the zone-based reinforcement learning method performs well for both provided demand and its variations (higher density, changes in client types). The application of proposed algorithms using real demand shows that the required fleet-size is significantly smaller, than the number of vehicles in provided real data (80 instead of 442).

7.8 Conclusion

In Section 7.3 we studied the dependency of the number of satisfied demands in the ring network from the battery size. The results obtained show not clearly monotonous nature of this dependency. This is caused by the occurring delays due to the chargings and consecutive impossibilities to satisfy arriving clients. In the complete information case, however, this dependency has more evident nature.

In Section 7.4 we provided the comparison between vehicle-based and station-based reinforcement learning methods, as well as with centralised station-based model. The results show, that the station-based method outperforms the vehicle-based one in terms of convergence time. Therefore, it provides better averaged performance characteristics within multiple iterations or, in other scenario setting, in long-term perspective. The centralised station-based RL model provides better results in long-term perspective, but with the price of bigger execution time.

Moreover we compared 2 classical methods for ride-sharing, searching on the way (SOTW) and searching in the area of the origin and/or destination (SOTA). We have shown that the SOTW method outperforms SOTA one on this network. However, it is achieved with the price of execution time.

In Sections 7.5 and 7.6, a set of empty vehicle redistribution algorithms for PRT or autonomous taxi services were investigated, from a passenger service and generalized cost perspectives.

The results presented show that the IBR based strategies perform well for provided demand, especially in combination with existing nearest neighbours strategies such as SNN and STN. The results for Scenario 4 show, that the STN+IBRTL redistribution strategy is one of the most robust and best performing for the problem with time windows.

The implementation of proposed algorithms to the real demand case of Scenario 4 shows that STN based strategies of empty vehicle redistribution show the best results in terms of the number of clients served within their constraints, and STN+IBR strategy shows a significant improvement of the maximal waiting time for the problem without time windows. In Scenario 4, we focus purely on the passenger service perspective, ignoring the cost of empty vehicle runs to the operator.

In Section 7.7 we investigated management methods of huge system. We provide numerically explicit analysis of the system performances for different methods and justify that reinforcement learning method with applied heuristics shows better convergence as well as better results in long run. Using computational experiments on the Porto city road network and real taxi data set for demand, we studied the system performances under different conditions and methods. One of obtained results allowed to study of dependency of number of zones in the system from number of served clients and processor execution time.

The presented results show that the zone-based reinforcement learning method performs well for provided demand, especially when combined with additional methods for EVR and ride-sharing. The application of proposed algorithms using real demand shows that these methods provide the best results in terms of the number of clients served within their constraints, and that well-suited zonal decomposition shows well-balanced relation between execution time and the number of satisfied demands.

Chapter 8

Conclusion and discussion

In this thesis we considered the problem of autonomous taxi system management. The considered taxi system aims to provide a certain quality of service, expressed in terms of satisfied client demands. The demand is considered to be satisfied if the main characteristics, such as passenger waiting times, their walking distance and shared ride preferences, are satisfied. The main objective is to design the reservation strategies allowing for a dynamic sharing of vehicle fleet by customers.

This optimisation problem reveals a certain number of objective functions. The first one is to maximise the number of satisfied clients with fixed taxi-fleet size. The second one is to optimally consume the energy, thus to minimise the duration of empty trips. The third one is to minimise the fleet size to serve a certain percentage of clients in the system.

In order to solve this problem, the following set of subproblems was formulated:

1. Charging management problem, where the objective is to reach an equilibrium between chargings and movements in the system.

2. Empty vehicle redistribution problem, which aims to manage idle vehicles in order to meet the demand.

3. Dynamic ride-sharing problem, which aims to reveal clients which can be served simultaneously by one vehicle. The basic idea is to identify the customers whose requests can be grouped and assigned to the same vehicle while guaranteeing a certain quality of service. The implementation of the ride-sharing allows to use the vehicle fleet more effectively, thus, to serve more clients with less vehicles.

In this work, we provide theoretical and experimental results.

In Chapter 2 we discussed the state-of-the-art in the taxi management problem, and have chosen the benchmarks and methods to use in this work.

In Chapter 3 we provided the mathematical and simulation model of the system, and formulated the objectives in terms of this model. We have introduced the notion of zone in the road graph network, which aims to provide the granularity in the system.

In Chapter 4 we provided the functional architecture of aTaxi system management. The main contributions of this chapter are the following. We provided heuristics of empty slot availability in charging and parking stations. We provided charging strategies that depend on overall system conditions. We developed new methods of road graph clustering, named Fill Clustering Method and Spill Clustering Method. These methods provide non-overlapping and overlapping uniform network clustering respectively. We investigated new methods of empty vehicle redistribution for problems with and without time windows. We discussed methods of ride-sharing in autonomous taxi system and provided new method of zone-based ride-sharing. We proposed new methods of zone-based empty vehicle redistribution, based on surplus/deficit and zone indexing respectively.

In Chapter 5 we focused on new reinforcement learning approach to the taxi management. We provided the learning at different levels. We introduced three decision levels, named microlevel, mesolevel and macrolevel, and provided main lines of the optimisation at these levels. The microlevel is divided into the vehicle-based and station-based reinforcement learning models. In the vehicle-based RL model, the memory of each vehicle is individual and vehicles do not share the information about the rewards and penalties obtained under different conditions. In the station-based RL model, the learning information is shared in the location, and vehicles located in the same area have the same learning information. The mesolevel is the zone-based reinforcement learning model. This method allows greater scalability under bigger granularity, thus, it is the basis for the real-time big taxi system management. The macrolevel provides centralised reinforcement learning models, both station-based and zone-based. These methods can be used both in the real-life systems as well as in the prediction analysis for taxi systems. The last section of this chapter provided methods for aTaxi system management in the complete information case in order to reveal the price of anarchy.

In Chapter 6 we have provided a new method of obtaining a consistent, real weighted road graph of a city, based on OpenStreetMap (OSM) network data and real vehicle movement GPS data sets. The proposed generic graph generation method is centered around finding a trajectory on the OSM graph close to the GPS data polyline of the trajectory, to create a graph which can be then used in computational models and in transport network analysis. The weight of every arc of such a graph has a clustered structure depending on the time of the day and the weekday, which is necessary for the aTaxi management optimisation purposes. The resulting graph allows obtaining velocity patterns on the arcs to predict vehicle movements. The rest of this chapter described the evaluation scenarii, including their road network graphs, client generations and vehicle characteristics.

In Chapter 7 we provided the evaluation of proposed methods on these scenarii. The convergence and performance of vehicle-based and station-based RL models were compared on different case studies. The purely heuristical classical methods of taxi management were compared with the proposed method on small networks. We provide the comparison between different methods, and show, that on both the real data of Porto network and synthetic ones, the method based on zone-based RL and using the heuristics for empty vehicle redistribution, ride-sharing and charging strategies shows the best performances on the long term. The evaluation on different client type distributions with different parameters is provided. The fleet-size influence on the system performances under different conditions was also studied. We illustrate our results on a series of experiments using simulations on synthetic data and classical benchmarks with many observations.

We believe that all these results open new work directions.

The first one is the search of the balance between reactive and proactive optimisations. The reactive optimisation aims to serve the maximal number of current demands and to use the total fleet in its maximal efficiency. In contrast, the proactive optimisation needs to reserve some of idle vehicles to serve the future demand, may be, more promising (to have more clients in one vehicle, to have longer and more expensive trips). The search of this equilibrium needs balancing between the best reactive strategies versus ones, which leave empty vehicles to be redistributed in the future.

The second one is to provide the analysis of taxi management strategies from the point of view of economics. We provided the generalised cost minimising objective function in the section of evaluation of Saclay network. However, the economical point of view requires to consider both profits of clients and the system as whole. The introduction of dynamic trip prices and benefits is one of the possible research directions.

The third direction is the more deep analysis of the proposed reinforcement learning methods. While we have provided some theoretical arguments (introduction of noisy convergence) and experimental results, a detailed theoretical analysis with explicit convergence rates would provide a better understanding of the benefits of the proposed methods.

Bibliography

[1] Andreas andgem. osmtoroadgraph website. https://github.com/AndGem/OsmToRoadGraph.

[2] A. Adler, D. Miculescu, and S. Karaman. Optimal policies for platooning and ride sharing in autonomy-enabled transportation. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016.

[3] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang. Sustainable passenger transportation: Dynamic ride-sharing. Technical report, Erasmus Research Institute of Management (ERIM), ERIM is the joint research, 2010.

[4] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro Atlanta. *Procedia-Social and Behavioral Sciences*, 17:532–550, 2011.

[5] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

[6] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.

[7] S. Algers, J. L. Dillen, S. Widlert, and et al. The national Swedish value of time study. In *PTRC European Transportation Forum, Warwick*, 1995.

[8] M. M. Alipour. A learning automata based algorithm for solving capacitated vehicle routing problem. *International Journal of Computer Science Issues (IJCSI)*, 9(2):138–145, 2012.

[9] A. M. R. Almeida, M. I. V. Lima, J. A. F. Macedo, and J. C. Machado. DMM: A Distributed Map-matching algorithm using the MapReduce Paradigm. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1706–1711. IEEE, 2016.

[10] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, pages 611–675, 2017.

[11] A. Amey, J. Attanucci, and R. Mishalani. Real-time ridesharing: opportunities and challenges in using mobile phone technology to improve rideshare services. *Transportation Research Record: Journal of the Transportation Research Board*, (2217):103–110, 2011.

[12] J. E. Anderson. Control of personal rapid transit systems. *Journal of advanced transportation*, 32(1):57–74, 1998.

[13] I. Andréasson. Quasi-optimum redistribution of empty PRT vehicles. In *Automated People Movers VI: Creative Access for Major Activity Centers*, pages 541–550. ASCE, 1997.

[14] I. Andréasson. Reallocation of empty personal rapid transit vehicles en route. *Transportation Research Record*: Journal of the Transportation Research Board*, (1838):36–41, 2003.

[15] I. Andréasson. Ride-sharing on PRT. In *Automated People Movers 2005: Moving to Mainstream*, pages 1–7. 2005.

[16] I. Andréasson. Operational strategies from personal to mass transit. In *Automated People Movers and Automated Transit Systems 2016*, pages 288–297. 2016.

[17] I. Andréasson and P. J. Muller. Ridesharing strategies for automated transit networks. 2015.

[18] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[19] M. H. G. Anthony and N. Biggs. *Computational learning theory*, volume 30. Cambridge University Press, 1997.

[20] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. pages 106–113. In Proceedings of the thirtieth annual ACM symposium on Theory of computing, ACM, May 1998.

[21] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[22] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562, 2004.

[23] I. Ashlagi, D. Monderer, and M. Tennenholtz. Learning equilibrium in resource selection games. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 18. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[24] R. Bai, J. Li, J. Atkin, and G. Kendall. A novel approach to independent taxi scheduling problem based on stable matching. *Journal of the Operational Research Society*, 65(10):1501–1510, 2014.

[25] R. Baldacci, V. Maniezzo, and A. Mingozzi. An exact method for the car pooling problem based on lagrangean column generatione. *Operations Research*, 52(3):422–439, 2004.

[26] P. Basu and T. D. C. Little. Wireless ad hoc discovery of parking meters. In *MobiSys Workshop on Applications of Mobile Embedded Systems (WAMES'04)*, 2004.

[27] G. J. Beaujon and M. A. Turnquist. A model for fleet sizing and vehicle allocation. *Transportation Science*, 25(1):19–45, 1991.

[28] G. S. Becker. A theory of the allocation of time. *The economic journal*, pages 493–517, 1965.

[29] M. Beckmann, Ch. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. 1956.

[30] S. Belhaiza, P. Hansen, and G. Laporte. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269–281, 2014.

[31] M. G. H. Bell and K. I. Wong. A rolling horizon approach to the optimal dispatching of taxis. In *Transportation and Traffic Theory. Flow, Dynamics and Human Interaction. 16th International Symposium on Transportation and Traffic Theory*, pages 629–648, 2005.

[32] M. Ben-Akiva and M. Bierlaire. Discrete choice methods and their applications to short term travel decisions. In *Handbook of transportation science*, pages 5–33. Springer, 1999.

[33] G. Berbeglia, J.-F. Cordeau, and G. Laporte. A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing*, 24(3):343–355, 2012.

[34] D. P. Bertsekas. Differential training of rollout policies. *Proceedings of the 35th Allerton Conference on Communication, Control, and Computing, Allerton Park IL*, 1997.

[35] D. Bertsimas, J. Tsitsiklis, et al. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[36] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.

[37] J. Biagioni and J. Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation research record*, 2291(1):61–71, 2012.

[38] J. Bijsterbosch and A. Volgenant. Solving the rectangular assignment problem and applications. *Annals of Operations Research*, 181(1):443–462, 2010.

[39] F. Bistaffa, A. Farinelli, G. Chalkiadakis, and S. D. Ramchurn. A cooperative game-theoretic approach to the social ridesharing problem. *Artificial Intelligence*, 246:86–117, 2017.

[40] M. Boiteux and L. Baumstark. *Transports: choix des investissements et coût des nuisances*. La documentation française, France. Commissariat général du plan, 2001.

[41] M. Börjesson and J. Eliasson. Experiences from the Swedish Value of Time study. *Transportation Research Part A: Policy and Practice*, 59:144–158, 2014.

[42] B. Boyacı, K. G. Zografos, and N. Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733, 2015.

[43] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[44] L. Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

[45] S. Breux and J. Diaz. *La ville intelligente: origine, définitions, forces et limites d'une expression polysémique*. Institut national de la recherche scientifique — Centre Urbanisation Culture, 2017.

[46] M. Bruglieri, A. Colorni, and A. Luè. The relocation problem for the one-way electric vehicle sharing. *Networks*, 64(4):292–305, 2014.

[47] R.J. Brunstad. The taxi market-excess capacity and insufficient supply. Technical report, Norwegian School of Economics and Business Administration-, 1991.

[48] R. Bruntrup, S. Edelkamp, S. Jabbar, and B. Scholz. Incremental map generation with GPS traces. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pages 574–579. IEEE, 2005.

[49] W. Burghout. A note on the number of replication runs in stochastic traffic simulation models. *Unpublished report, Stockholm: Centre for Traffic Research*, 2004.

[50] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 38(2):156–172, 2008.

[51] M. Caliskan, A. Barthels, B. Scheuermann, and M. Mauve. Predicting parking lot occupancy in vehicular ad hoc networks. In *2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring*, pages 277–281. IEEE, 2007.

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

[53] A. M. Campbell and M. W. P. Savelsbergh. A decomposition approach for the inventory-routing problem. *Transportation science*, 38(4):488–502, 2004.

[54] Y. Cao, W. Yu, W. Ren, and G. Chen. An overview of recent progress in the study of distributed multi-agent coordination. *https://arxiv.org/pdf/1207.3231.pdf*, 2012.

[55] J. I. Castillo-Manzano and A. Sánchez-Braza. An evaluation of the establishment of a taxi flat rate from city to airport: the case of Seville. *Urban Studies*, 48(9):1909–1924, 2011.

[56] B. Caulfield. Estimating the environmental benefits of ride-sharing: A case study of Dublin. *Transportation Research Part D: Transport and Environment*, 14(7):527–531, 2009.

[57] M. Cebecauer, E. Jenelius, and W. Burghout. Integrated framework for real-time urban network travel time prediction on sparse probe data. *IET Intelligent Transport Systems*, 12(1):66–74, 2017.

[58] N. D. Chan and S. A. Shaheen. Ridesharing in north america: Past, present, and future. *Transport Reviews*, 32(1):93–112, 2012.

[59] H.-W. Chang, Y.-Ch. Tai, and J. Y.-J. Hsu. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):3–18, 2009.

[60] X. (M.) Chen, M. Zahiri, and Sh. Zhang. Understanding ridesplitting behavior of on-demand ride services: An ensemble learning approach. *Transportation Research Part C: Emerging Technologies*, 76:51–70, March 2017.

[61] R. Cheu, J. Xu, A. Kek, W. Lim, and W. Chen. Forecasting Shared-Use Vehicle Trips with Neural Networks and Support Vector Machines. *Transportation research record*, 1968(1):40–46, 2006.

[62] N. Chiabaut and C. Buisson. Replications in stochastic traffic flow models: incremental method to determine sufficient number of runs. In *Traffic and Granular Flow'07*, pages 35–44. Springer, 2009.

[63] C. G. Chorus, T. A. Arentze, and H. J. P. Timmermans. A random regret-minimization model of travel choice. *Transportation Research Part B: Methodological*, 42(1):1–18, 2008.

[64] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 67–73. ACM, 2005.

[65] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 9–21. ACM, 2016.

[66] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

[67] D. Costa. Performance and design of taxi services at airport passenger terminals. *Lisbon: Master in CTIS, IST*, 2009.

[68] N. Cressie. Statistics for spatial data. *Terra Nova*, 4(5):613–617, 1992.

[69] X. Dai, Ch.-K. Li, and A.B. Rad. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Trnsportations Systems*, 6:285–293, September 2005.

[70] A. Daly, F. Tsang, and Ch. Rohr. The value of small time savings for non-business travel. *Journal of Transport Economics and Policy (JTEP)*, 48(2):205–218, 2014.

[71] M. S. Daskin. What you should know about location modeling. *Naval Research Logistics (NRL)*, 55(4):283–294, 2008.

[72] W. B. Daszczuk, W. Choromański, J. Mieścicki, and W. Grabski. Empty vehicles management as a method for reducing passenger waiting time in personal rapid transit networks. *IET Intelligent Transport Systems*, 9(3):231–239, 2014.

[73] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio. Artificial neural networks applied to taxi destination prediction. *https://arxiv.org/pdf/1508.00021.pdf*, 2015.

[74] E. Deakin, K. Frick, and K. Shively. Markets for dynamic ridesharing? Case of Berkeley, California. *Transportation Research Record: Journal of the Transportation Research Board*, (2187):131–137, 2010.

[75] T. DeMent, A. Eisen, Ch. Riter, J. Thornton, Dr. Little, and Dr. Thomas. "ParkSens – a Parking Space Locator System". http://www.parksens.com/.

[76] G. Desaulniers, F. Errico, S. Irnich, and M. Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016.

[77] A. C. DeSerpa. A theory of the economics of time. *The Economic Journal*, 81(324):828–846, 1971.

[78] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[79] R. Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173, 2005.

[80] K. W. Douglas. "Truly" Empty Vehicle Repositioning and Fleet-Sizing: Optimal Management of an Autonomous Taxi System in New Jersey on a Typical Weekday, 2015.

[81] J. Dowling and S. Haridi. Decentralized reinforcement learning for the online optimization of distributed systems. In *Reinforcement Learning*. IntechOpen, 2008.

[82] K. Epperson. Santa Barbara Dynamic Ridesharing Pilot Program, 2015.

[83] S. Erdoğan and E. Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, 2012.

[84] A. W. Evans. On the theory of the valuation and allocation of time. *Scottish Journal of Political Economy*, 19(1):1–17, 1972.

[85] I.-A. F. Sætermo F. Ramjerdi, L. Rand and K. Sælensminde. The Norwegian Value of Time Study Part I. *Institute of Transport Economics, Oslo*, 1997.

[86] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure nash equilibria. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 604–612. ACM, 2004.

[87] D. J. Fagnant and K. M. Kockelman. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies*, 40:1–13, 2014.

[88] D. J. Fagnant and K. M. Kockelman. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas. *Transportation*, pages 1–16, 2016.

[89] W. Fan, R. Machemehl, and N. Lownes. Carsharing: Dynamic decision-making problem for vehicle allocation . *Transportation Research Record: Journal of the Transportation Research Board*, (2063):97–104, 2008.

[90] E. Fatnassi, O. Chebbi, and J. Chaouachi. Dealing with the Empty Vehicle Movements in Personal Rapid Transit System with Batteries Constraints in a Dynamic Context. *Journal of Advanced Transportation*, 2017.

[91] E. Fatnassi, O. Chebbi, and J. Ch. Siala. Two strategies for real time empty vehicle redistribution for the personal rapid transit system. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 1888–1893. IEEE, 2013.

[92] A. Febbraro, N. Sacco, and M. Saeednia. One-way carsharing: solving the relocation problem. *Transportation Research Record: Journal of the Transportation Research Board*, (2319):113–120, 2012.

[93] Á. Felipe, M. T. Ortuño, G. Righini, and G. Tirado. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111–128, 2014.

[94] G. M. Fetene, S. Kaplan, S. L. Mabit, A. F. Jensen, and C. G. Prato. Harnessing big data for estimating the energy consumption and driving range of electric vehicles. *Transportation Research Part D: Transport and Environment*, 54:1–11, 2017.

[95] M. Fischetti, P. Toth, and D. Vigo. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42(5):846–859, 1994.

[96] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and Sh. Whiteson. Counterfactual multi-agent policy gradients. *https://arxiv.org/pdf/1705.08926.pdf*, 2017.

[97] M. Fosgerau, K. Hjorth, and S. V. Lyk-Jensen. The Danish value of time study. *Danmarks Transport Forskning Report*, 2007.

[98] L. Foti, J. Lin, O. Wolfson, and N. D. Rishe. The nash equilibrium among taxi ridesharing partners. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 72. ACM, 2017.

[99] Luca Foti, Jane Lin, and Ouri Wolfson. Optimum versus nash-equilibrium in taxi ridesharing. *GeoInformatica*, pages 1–29, 2019.

[100] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12:133–137, 1981.

[101] T. Franke and J. F. Krems. Understanding charging behaviour of electric vehicle users. *Transportation Research Part F: Traffic Psychology and Behaviour*, 21:75–89, 2013.

[102] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[103] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artifical Intelligence*, 14(5):771–780, September 1999.

[104] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, November 2001.

[105] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

[106] K. Ghoseiri, A. E. Haghani, and M. Hamedi. *Real-time rideshare matching problem.* Mid-Atlantic Universities Transportation Center, Berkeley, 2011.

[107] Sh. Ghosh, K. Page, and D. De Roure. An application of network lasso optimization for ride sharing prediction. *arXiv preprint arXiv:1606.03276*, 2016.

[108] R. Gilbert and A. Perl. Grid-connected vehicles as the core of future land-based transport systems. *Energy Policy*, 35(5):3053–3060, 2007.

[109] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[110] F. Glover. Tabu search: Part II. *ORSA Journal on computing*, 2(1):4–32, 1990.

[111] P. Goel, L. Kulik, and K. Ramamohanarao. Optimal pick up point selection for effective ride sharing. *IEEE Transactions on Big Data*, 3(2):154–168, 2017.

[112] B. L. Golden and C. C. Skiscim. Using simulated annealing to solve routing and location problems. *Naval Research Logistics Quarterly*, 33(2):261–279, 1986.

[113] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[114] J. S. Greenfeld. Matching GPS observations to locations on a digital map. In *81th annual meeting of the transportation research board*, volume 1, pages 164–173. Washington, DC, 2002.

[115] S. Grover, N. Gupta, and A. Panchol. Constant factor approximation algorithms for uniform hard capacitated facility location problems: Natural lp is not too bad. *arXiv preprint arXiv:1606.08022*, 2016.

[116] K. Guan, J. Kakande, and J. Cho. On deploying encryption solutions to provide secure transport-as-a-service (TaaS) in core and metro networks. In *ECOC 2016; 42nd European Conference on Optical Communication*, pages 1–3. VDE, 2016.

[117] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248, 1999.

[118] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[119] J. P. Hanna, M. Albert, D. Chen, and P. Stone. Minimum cost matching for autonomous carsharing. *IFAC-PapersOnLine*, 49(15):254–259, 2016.

[120] P. Hansen, J. Henderson, M. Labbe, J. Peeters, and J. Thisse. *Systems of cities and facility location*. Taylor & Francis, 2013.

[121] R. Hendershot. Morgantown Personal Rapid Transit System, System Operation Description Manual. Technical report, 1975.

[122] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018, 2016.

[123] B. Hillier and J. Hanson. *The social logic of space*. Cambridge university press, 1989.

[124] G. E. Hinton, T. J. Sejnowski, and T. A. Poggio. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[125] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[126] C. Hosage and M. Goodchild. Discrete space location-allocation solutions from genetic algorithms. *Annals of Operations Research*, 6(2):35–46, 1986.

[127] D. Hsu and M. Telgarsky. Greedy bi-criteria approximations for k-medians and k-means. 2016. arXiv preprint arXiv:1607.06203.

[128] J. Huang, M. Deng, J. Tang, Sh. Hu, H. Liu, S. Wariyo, and J. He. Automatic generation of road maps from low quality gps trajectory data via structure learning. *IEEE Access*, 6:71965–71975, 2018.

[129] L. Hultkrantz, G. Lindberg, and Ch.-Zh. Li. *Some problems in the consumer preference approach to multimodal transport planning*. Centrum för transport-och samhällsforskning, 1996.

[130] L. Hultkrantz and R. Mortazavi. Anomalies in the value of travel-time changes. *Journal of Transport Economics and Policy (JTEP)*, 35(2):285–299, 2001.

[131] R. Z. Hwang, R. C. T. Lee, and R. C. Chang. The slab dividing approach to solve the Euclidean p-center problem. *Algorithmica*, 9:1–22, 1993.

[132] Sh. H. Jacobson and D. M. King. Fuel saving and ridesharing in the US: Motivations, limitations, and opportunities. *Transportation Research Part D: Transport and Environment*, 14(1):14–21, 2009.

[133] S. R. Jara-Díaz. Allocation and valuation of travel time savings. *Handbook of Transport Modelling*, 2:363–379, 2008.

[134] X. Jin, B. P. Carlin, and S. Banerjee. Generalized hierarchical multivariate car models for areal data. *Biometrics*, 61(4):950–961, 2005.

[135] M. B. Johnson. Travel time and the price of leisure. *Economic Inquiry*, 4(2):135–145, 1966.

[136] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[137] Kaggle Chicago Data. https://www.kaggle.com/chicago/chicago-taxi-trips.

[138] Kaggle Chicago Taxi Rides Data. Chicago Taxi Rides 2016, 2016. https://www.kaggle.com/chicago/chicago-taxi-rides-2016.

[139] Kaggle Data Source. Your Home for Data Science. http://www.kaggle.com/.

[140] Kaggle New York City Taxi Data. New York City Taxi with OSRM. https://www.kaggle.com/oscarleo/new-york-city-taxi-with-osrm.

[141] Kaggle New York Data. Uber Pickups in New York City. https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city.

[142] Kaggle Taxi Time Prediction Data. ECML/PKDD 15: Taxi Trip Time Prediction (II). https://www.kaggle.com/c/pkdd-15-taxi-trip-time-prediction-ii.

[143] Kaggle Taxi Trajectory Prediction Data. ECML/PKDD 15: Taxi Trajectory Prediction (I). https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i.

[144] L. Kaixiang, Z. Renyu, X. Zhe, and Z. Jiayu. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1774–1783, 2018.

[145] E. Kamar and E. Horwitz. Collaboration and shared plans in the open world: Studies of ridesharing. *the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

[146] P. Kandal. Procedure for determining the probability of finding a parking place, April 16 2013. US Patent 8,423,275.

[147] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 89–98. ACM, 2012.

[148] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.

[149] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. I: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

[150] M. G. Karlaftis and E. L. Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, pages 387–399, 2011.

[151] M. Kearns. Thoughts on hypothesis boosting. *Unpublished manuscript*, 45:105, 1988.

[152] M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[153] A. Kek, R. Cheu, and M. Chor. Relocation simulation model for multiple-station shared-use vehicle systems. *Transportation Research Record: Journal of the Transportation Research Board*, (1986):81–88, 2006.

[154] A. Kleiner, B. Nebel, and V. A. Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. In *Twenty-Second International Joint Conference on Artificial Intelligence*, volume 11, pages 266–272, 2011.

[155] K. Kockelman, S. Boyles, P. Stone, D. Fagnant, R. Patel, M. W. Levin, G. Sharon, M. Simoni, M. Albert, H. Fritz, and et al. An assessment of autonomous vehicles: Traffic impacts and infrastructure needs—final report. Technical report, 2018.

[156] A. Kornhauser, A. Chang, C. Clark, J. Gao, D. Korac, B. Lebowitz, and A. Swoboda. Uncongested mobility for all: New Jersey's area-wide aTaxi system. *Princeton University: Princeton, NJ, USA*, 2013.

[157] Kornhauser, A. Smart Driving Cars website. http://www.smartdrivingcar.com/.

[158] Kornhauser, A. Theses under supervision de Alain Kornhauser.

[159] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[160] P. Kumar and P. Kumar. Almost optimal solutions to k-clustering problems. *International Journal of Computational Geometry & Applications*, 20:431–447, 2010.

[161] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: Utility functions, random losses and ecn marks. *IEEE/ACM Transactions on Networking (TON)*, 11(5):689–702, 2003.

[162] A. A. Kurzhanskiy and P. Varaiya. Active traffic management on road networks: a macroscopic approach. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4607–4626, 2010.

[163] L. Kuyer, Sh. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer, 2008.

[164] R. Lagos. An alternative approach to search frictions. *Journal of Political Economy*, 108(5):851–873, 2000.

[165] R. Lagos. An analysis of the market for taxicab rides in new york city. *International Economic Review*, 44(2):423–434, 2003.

[166] G. Laporte, H. Mercure, and Y. Nobert. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46, 1986.

[167] J. Lees-Miller, J. Hammersley, and N. Davenport. Ride sharing in personal rapid transit capacity planning. In *Automated People Movers 2009: Connecting People, Connecting Places, Connecting Modes*, pages 321–332. 2009.

[168] J. Lees-Miller, J. Hammersley, and R. Wilson. Theoretical maximum capacity as benchmark for empty vehicle redistribution in personal rapid transit. *Transportation Research Record: Journal of the Transportation Research Board*, (2146):76–83, 2010.

[169] J. D. Lees-Miller. Minimising average passenger waiting time in personal rapid transit systems. *Annals of Operations Research*, 236(2):405–424, 2016.

[170] J. D. Lees-Miller and R. E. Wilson. Proactive empty vehicle redistribution for personal rapid transit and taxis. *Transportation Planning and Technology*, 35(1):17–30, 2012.

[171] N. Lenoir and I. Laplace. Beyond traditional value-of-time: passenger behavior for multimodal door-to-door travels in the age of information technologies. In *European Transport Conference 2017*, 2017.

[172] M. W. Levin, K. Kockelman, S. D. Boyles, and T. Li. A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application. *Computers, Environment and Urban Systems*, 64:373–383, 2017.

[173] A. Levofsky and A. Greenberg. Organized dynamic ride sharing: The potential environmental benefits and the opportunity for advancing the concept. In *Transportation Research Board 2001 Annual Meeting*, pages 7–11, 2001.

[174] T. Lewinson and Sh. Lewinson. Method of selective ride-sharing among multiple users along an optimized travel route, April 16 2007. US Patent App. 11/735,552.

[175] S. Li. A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem. *Automata, Languages and Programming*, 6756:77–88, 2011.

[176] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. pages 1774–1783, 2018.

[177] W.-H. Lin and J. Zeng. Experimental study of real-time bus arrival time prediction with GPS data. *Transportation Research Record*, 1666(1):101–109, 1999.

[178] Ye. Lin, W. Li, F. Qiu, and H. Xu. Research on optimization of vehicle routing problem for ride-sharing taxi. *Procedia-Social and Behavioral Sciences*, 43:494–502, 2012.

[179] T. E. Lisco. Value of commuters travel time-a study in urban transportation. Technical report, 1968.

[180] N. Liu, M. Liu, J. Cao, G. Chen, and W. Lou. When Transportation Meets Communication: V2P over VANETs. Technical report, Hong Kong RGC GRF grant, 2010.

[181] M. Longo, D. Zaninelli, F. Viola, P. Romano, R. Miceli, M. Caruso, and F. Pellitteri. Recharge stations: A review. In *2016 Eleventh International Conference on Ecological Vehicles and Renewable Energies (EVER)*, pages 1–8. IEEE, 2016.

[182] R. Lowe, Y. Wu, A. Tamar, J. Harb, A. Pieter, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6382–6393. Curran Associates Inc., 2017.

[183] S. Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013.

[184] Sh. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service . In *Data Engineering (ICDE)*, pages 410–421. IEEE, 2013.

[185] Sh. Ma, Y. Zheng, and O. Wolfson. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1782–1795, 2015.

[186] M. Maciejewski and K. Nagel. Towards multi-agent simulation of the dynamic vehicle routing problem in matsim. In *International Conference on Parallel Processing and Applied Mathematics*, pages 551–560. Springer, 2011.

[187] P. J. Mackie, M. Wardman, A. S. Fowkes, G. Whelan, J. Nellthorp, and J. Bates. Values of travel time savings UK. 2003.

[188] K. Makarychev, Y. Makarychev, M. Sviridenko, and J. Ward. A bi-criteria approximation algorithm for $k-$means. 2015. arXiv preprint arXiv:1507.04227.

[189] L. Mariella and M. Tarantino. Spatial temporal conditional auto-regressive model: A new autoregressive matrix. *Austrian Journal of statistics*, 39(3):223–244, 2010.

[190] A. Martínez-Usó, J. Mendes-Moreira, L. Moreira-Matias, M. Kull, and N. Lachiche. Discovery challenges co-located with european conference on machine learning and principles and practice of knowledge discovery in databases. 2015.

[191] N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1:194–197, 1982.

[192] D. Meignan, J.-C. Créput, and A. Koukam. A coalition-based metaheuristic for the vehicle routing problem. In *Evolutionary Computation*, pages 1176–1182. IEEE, 2008.

[193] R. Meir, M. Tennenholtz, Y. Bachrach, and P. Key. Congestion games with agent failures. In *Twenty-sixth AAAI conference on artificial intelligence*, 2012.

[194] S. Melkote and M. S. Daskin. Capacitated facility location/network design problems. *European Journal of Operational Research*, 129:481–495, 2001.

[195] N. Merat, R. Madigan, and S. Nordhoff. Human factors, user requirements, and user acceptance of ride-sharing in automated vehicles. In *International Transport Forum Roundtable on Cooperative Mobility Systems and Automated Driving*, pages 1–28, 2016.

[196] R. S. Michalski. Inferential theory of learning: Developing foundations for multistrategy learning. Technical report, 1992.

[197] W. Min and L. Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19:606–616, August 2011.

[198] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939, 2007.

[199] H. Mohring, J. Schroeter, and P. Wiboonchutikula. The values of waiting time, travel time, and a seat on a bus. *The RAND Journal of Economics*, pages 40–56, 1987.

[200] M. Montazery and N. Wilson. Learning user preferences in matching for ridesharing. Technical report, Insight Centre for Data Analytics, School of Computer Science and IT University College Cork, Ireland, 2016.

[201] D. T. Mortensen and C. A. Pissarides. Job creation and job destruction in the theory of unemployment. *The review of economic studies*, 61(3):397–415, 1994.

[202] T. Murray and F. Steele. System and method for facilitating ridesharing murray2003system, June 19 2003. US Patent App. 10/464,737.

[203] A. Musrrat, P. Millie, and A. Ajith. Simplex differential evolution. *Acta Polytechnica Hungarica*, 6(5), 2009.

[204] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[205] M. Ojeda-Cabral, R. Batley, and S. Hess. The value of travel time: random utility versus random valuation. *Transportmetrica A: Transport Science*, 12(3):230–248, 2016.

[206] C. J. Oort. The evaluation of travelling time. *Journal of transport economics and policy*, pages 279–286, 1969.

[207] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[208] D. Papadimitriou and D. Colle. Mixed-integer optimization for the combined capacitated facility location-routing problem. volume 73, pages 37–62. Springer, 2018.

[209] J. Pearl. The solution for the branching factor of the alpha-beta pruning algorithm and its optimality. *Communications of the ACM*, 25(8):559–564, 1982.

[210] M. E. H. Pedersen. Good parameters for differential evolution. Technical report, Hvass Laboratories, 2010. HL1002.

[211] M. E. H. Pedersen. Good parameters for particle swarm optimization. Technical report, Hvass Laboratories, 2010. HL1001.

[212] B. Petrongolo and C. A. Pissarides. Looking into the black box: A survey of the matching function. *Journal of Economic literature*, 39(2):390–431, 2001.

[213] T.-H. Pham, G. De Magistris, and R. Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. *arXiv preprint arXiv:1709.07643*, 2017.

[214] S. Porta, P. Crucitti, and V. Latora. The network analysis of urban streets: a primal approach. *Physica A: Statistical Mechanics and its Applications*, 33:705–725, 2006.

[215] S. Porta, P. Crucitti, and V. Latora. The network analysis of urban streets: A dual approach. *Physica A: Statistical Mechanics and its Applications*, 369:853–866, September 2006.

[216] W. B. Powell and R. K.-M. Cheung. A network recourse decomposition method for dynamic networks with random arc capacities. *Networks*, 24(7):369–384, 1994.

[217] W. B. Powell and H. Topaloglu. Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10:555–635, 2003.

[218] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization.* Springer, 2005.

[219] S. Proper and P. Tadepalli. Scaling model-based average-reward reinforcement learning for product delivery link. In *European Conference on Machine Learning*, pages 735–742. Springer, 2006.

[220] X. Qian, S. V. Ukkusuri, C. Yang, and F. Yan. Forecasting short-term taxi demand using boosting-GCRF. *6th ACM SIGKDD International Workshop on Urban Computing*, August 2017.

[221] M. Quddus and S. Washington. Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies*, 55:328–339, 2015.

[222] M. A. Quddus. *High integrity map matching algorithms for advanced transport telematics applications.* PhD thesis, Imperial College London, 2006.

[223] J. Quinn. System and method for predicting parking spot availability, February 28 2008. US Patent App. 11/849,493.

[224] M. Rahmani. *Path inference of sparse gps probes for urban networks: Methods and applications.* PhD thesis, KTH Royal Institute of Technology, 2012.

[225] M. Rahmani, H. N. Koutsopoulos, and E. Jenelius. Travel time estimation from sparse floating car data with consistent path inference: A fixed point approach. *Transportation Research Part C: Emerging Technologies*, 85:628–643, 2017.

[226] S. Ringbom and O. Shy. Advance booking, cancellations, and partial refunds. *Economics Bulletin*, 13(1):1–7, 2004.

[227] R. Rogerson, R. Shimer, and R. Wright. Search-theoretic models of the labor market: A survey. *Journal of economic literature*, 43(4):959–988, 2005.

[228] R. Roor, M. Karg, A. Liao, W. Lei, and A. Kirsch. Predictive ridesharing based on personal mobility patterns. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1322–1329. IEEE, 2017.

[229] R. W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[230] T. Roughgarden and É. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and economic behavior*, 47(2):389–403, 2004.

[231] I. Saadi, M. Wong, B. Farooq, J. Teller, and M. Cools. An investigation into machine learning approaches for forecasting spatio-temporal demand in ride-hailing service. *arXiv:1703.02433v1*, pages 1–31, March 2017.

[232] D. O. Santos and E. C. Xavier. Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, 42(19):6728–6737, 2015.

[233] P. S. Sastry, V. V. Phansalkar, and M. A. L. Thathachar. Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(5):769–777, May 1994.

[234] R. Schakenbos, L. La Paix, S. Nijenstein, and K. T. Geurs. Valuation of a transfer in a multimodal public transport trip. *Transport policy*, 46:72–81, 2016.

[235] M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

[236] S. Schönfelder, K. W. Axhausen, N. Antille, and M. Bierlaire. Exploring the potentials of automatically collected GPS data for travel behaviour analysis: A Swedish data source. Technical report, 2002.

[237] J. R. Schroeter. A model of taxi service under fare structure and fleet size regulation. *The Bell Journal of Economics*, pages 81–96, 1983.

[238] N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27(11-12):1201–1225, 2000.

[239] N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

[240] J. Shi, Y. Gao, and N. Yu. Routing electric vehicle fleet for ride-sharing. In *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–6. IEEE, 2018.

[241] W. Shi, Sh. Shen, and Y. Liu. Automatic generation of road network map from massive GPS, vehicle trajectories. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–6. IEEE, 2009.

[242] R. Sirisoma, S. C. Wong, W. Lam, D. Wang, H. Yang, and P. Zhang. Empirical evidence for taxi customer-search model. In *Proceedings of the Institution of Civil Engineers-Transport*, volume 163, pages 203–210. Thomas Telford Ltd, 2010.

[243] K. A. Small. Valuation of travel time. *Economics of transportation*, 1(1-2):2–14, 2012.

[244] R. L. Smith. Turnpike results for single location capacity expansion. *Management Science*, 25(5):474–484, 1979.

[245] B. Sniezynski, W. Wojcik, J. D. Gehrke, and J. Wojtusiak. Combining rule induction and reinforcement learning: An agent-based vehicle routing. In *Machine Learning and Applications*, pages 851–856. IEEE, 2010.

[246] D.-P. Song and C. F. Earl. Optimal empty vehicle repositioning and fleet-sizing for two-depot service systems. *European Journal of Operational Research*, 185(2):760–777, 2008.

[247] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisara. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review*, 91:190–207, July 2016.

[248] R. Storn and K. Price. *Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces*, volume 11. 1997.

[249] Y. Sun, J. Huang, Y. Chen, R. Zhang, and X. Du. Location selection for utility maximization with capacity constraints. pages 2154–2158, 2012.

[250] Z. Sun and X. J. Ban. Vehicle classification using GPS data. *Transportation Research Part C: Emerging Technologies*, 37:102–117, 2013.

[251] S. Suri, C. D. Tóth, and Y. Zhou. Selfish load balancing and atomic congestion games. *Algorithmica*, 47(1):79–96, 2007.

[252] E. Suryani, Sh.-Y. Chou, and Ch.-H. Chen. Air passenger demand forecasting and passenger terminal capacity expansion: A system dynamics framework. *Expert Systems with Applications*, 37(3):2324–2339, 2010.

[253] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. 2(4), 1998.

[254] N. Ta, G. Li, T. Zhao, J. Feng, H. Ma, and Z. Gong. An efficient ride-sharing framework for maximizing shared route. *IEEE Transactions on Knowledge and Data Engineering*, 30(2):219–233, 2017.

[255] E.-G. Talbi. Metaheuristics: from design to implementation. 74, 2009.

[256] D. Tamagawa, E. Taniguchi, and T. Yamada. Evaluating city logistics measures using a multi-agent model. *Procedia-Social and Behavioral Sciences*, 2(3):6002–6012, 2010.

[257] M. Tamari. An optimal parking problem. *Journal of Applied Probability*, 19(4):803–814, 1982.

[258] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. *arXiv preprint arXiv:1511.08779*, 2015.

[259] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[260] Ph. Thaithatkul, T. Seo, T. Kusakabe, and Ya. Asakura. Simulation approach for investigating dynamics of passenger matching problem in smart ridesharing system. volume 21, pages 29–41. Elsevier, 2017.

[261] M. A. L. Thathachar and P. S. Sastry. Varieties of learning automata: an overview. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 32(6):711–722, 2002.

[262] Rijkswaterstaat The Hague Consulting Group for AVV. The second Netherlands' value of time study: final report. 1998.

[263] D. C. Thorns. *The transformation of cities: urban theory and urban life*. Macmillan International Higher Education, 2017.

[264] P. Toth and D. Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1-3):487–512, 2002.

[265] C. Unsal, P. Kachroo, and J. S. Bay. Simulation study of multiple intelligent vehicle control using stochastic learning automata. *Transactions of the Society for Computer Simulation*, 14(4):193–210, 1997.

[266] M. van der Heijden, M. Ebben andb N. Gademann, and A. van Harten. Scheduling vehicles in automated transportation systems algorithms and case study. *OR spectrum*, 24(1):31–58, 2002.

[267] M. van Gerven and S. Bohte. *Artificial neural networks as models of neural information processing*, volume 11. Frontiers, 2017.

[268] A. Vasiliev and D. Tarkhov. *Neural network modelling. Principles. Algorithms. Applications (in Russian)*. Publishing house of Polytechnic University, Saint-Petersburg, 2009.

[269] V. Verter and M. C. Dincer. An integrated evaluation of facility location, capacity acquisition, and technology selection for designing global manufacturing strategies. *European Journal of Operational Research*, 60(1):1–18, 1992.

[270] Th. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.

[271] H. Wang and R. L. Cheu. Operations of a taxi fleet for advance reservations using electric vehicles and charging stations. *Transportation Research Record*, 2352(1):1–10, 2013.

[272] H. Wang, R. L. Cheu, and D.-H. Lee. Intelligent taxi dispatch system for advance reservations. *Journal of Public Transportation*, 17(3):8, 2014.

[273] H. Wang, D.-H. Lee, and R. Cheu. PDPTW based taxi dispatch modeling for booking service. In *2009 Fifth International Conference on Natural Computation*, volume 1, pages 242–247. IEEE, 2009.

[274] X. Wang, N. Agatz, and A. Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 52(4):850–867, 2017.

[275] X. Wang, M. Dessouky, and F. Ordonez. A pickup and delivery problem for ridesharing considering congestion. *Transportation letters*, 8(5):259–269, 2016.

[276] Y. Wang, Y. Zhu, Zh. He, Y. Yue, and Q. Li. Challenges and opportunities in exploiting large-scale gps probe data. *HP Laboratories, Technical Report HPL-2011-109*, 21, 2011.

[277] M. Wardman. The value of travel time: a review of British evidence. *Journal of transport economics and policy*, pages 285–316, 1998.

[278] C. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[279] C. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[280] B. Watters. Values of travel time savings used in road project evaluation: a cross-country/jurisdiction comparison. In *Papers of the Australasian Transport Research Forum October 1992, Canberra, Volume 17, Part 1*, volume 17, 1992.

[281] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

[282] C. Whelan, G. Harrell, and J. Wang. Understanding the k-medians problem. In *Proceedings of the International Conference on Scientific Computing (CSC)*, page 219. The Steering Committee of The World Congress in Computer Science, Computer, 2015.

[283] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation research part c: emerging technologies*, 8(1-6):91–108, 2000.

[284] MA Wiering. Multi-agent reinforcement learning for traffic light control link. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000.

[285] S. Winter and S. Nittel. Ad hoc shared-ride trip planning by mobile geosensor networks winter2006ad. *International Journal of Geographical Information Science*, 20(8):899–916, 2006.

[286] J. Wojtusiak, R. S. Michalski, K. A. Kaufman, and J. Pietrzykowski. The AQ21 natural induction program for pattern discovery: initial version and its novel features. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 523–526. IEEE, 2006.

[287] J. Wojtusiak, T. Warden, and O. Herzog. Machine learning in agent-based stochastic simulation: Inferential theory and evaluation in transportation logistics. *Computers & Mathematics with Applications*, 64:3658–3665, December 2012.

[288] D. H. Wolpert. Stacked generalization. Technical Report 2, 1992.

[289] K. I. Wong, S. C. Wong, M. Bell, and H. Yang. Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing markov chain approach. *Journal of advanced transportation*, 39(1):81–104, 2005.

[290] K. I. Wong, S. C. Wong, and H. Yang. Modeling urban taxi services in congested road networks with elastic demand. *Transportation Research Part B: Methodological*, 35(9):819–842, 2001.

[291] K. I. Wong, S. C. Wong, H. Yang, and J. Wu. Modeling urban taxi services with multiple user classes and vehicle modes. *Transportation Research Part B: Methodological*, 42(10):985–1007, 2008.

[292] Y. Z. Wong, D. A. Hensher, and C. Mulley. Emerging transport technologies and the modal efficiency framework: a case for mobility as a service (MaaS). In *International Conference on Competition and Ownership in Land Passenger Transport, 15th, 2017, Stockholm, Sweden*, 2017.

[293] X. Wu and D.-H. Lee. An integrated taxi dispatching strategy handling both current and advance bookings. *Journal of the Eastern Asia Society for Transportation Studies*, 10:1836–1855, 2013.

[294] X. Xing, T. Warden, T. Nicolai, and O. Herzog. SMIZE: a spontaneous ride-sharing system for individual urban transit. In *German Conference on Multiagent System Technologies*, pages 165–176. Springer, 2009.

[295] J. Xu, S. C. Wong, H. Yang, and Ch.-O. Tong. Modeling level of urban taxi services using neural network. *Journal of Transportation Engineering*, 125(3):216–223, 1999.

[296] T. Yamaguchi, Y. Tanaka, and M. Yachida. Speed up reinforcement learning between two agents with adaptive mimetism. In *Intelligent Robots and Systems*, volume 2, pages 594–600. IEEE, 1997.

[297] X. W. Yan, Z. D. Deng, and Z. Q. Sun. Competitive takagi-sugeno fuzzy reinforcement learning. In *Control Applications, 2001.(CCA'01). Proceedings of the 2001 IEEE International Conference on*, pages 878–883. IEEE, 2001.

[298] H. Yang, C. W. Y. Leung, S. C Wond, and M. G. H. Bell. Equilibria of bilateral taxi–customer searching and meeting on networks. *Transportation Research Part B: Methodological*, 44:1067–1083, September-November 2010.

[299] H. Yang and S. C. Wong. A network model of urban taxi services. *Transportation Research Part B: Methodological*, 32(4):235–246, 1998.

[300] H. Yang, S. C. Wong, and K. I. Wong. Demand–supply equilibrium of taxi services in a network under competition and regulation. *Transportation Research Part B: Methodological*, 36(9):799–819, 2002.

[301] H. Yang and T. Yang. Equilibrium properties of taxi markets with search frictions. *Transportation Research Part B: Methodological*, 45(4):696–713, 2011.

[302] H. Yang, M. Ye, W. H. Tang, and S. C. Wong. Regulating taxi services in the presence of congestion externality. *Transportation Research Part A: Policy and Practice*, 39(1):17–40, 2005.

[303] N. E. Young. k-medians, facility location, and the Chernoff-Wald bound. *arXiv preprint cs/0205047*, 2002.

[304] Y. Zhang and A. Haghani. A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 58 Part B:308–324, September 2015.

[305] X. Zheng, X. Liang, and K. Xu. Where to wait for a taxi? In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pages 149–156. ACM, 2012.

[306] H. Zhu and Y. Shi. Brain storm optimization algorithms with k-medians clustering algorithms. pages 107–110. Seventh International Conference, IEEE, March 2015.

[307] Sh. Zhu and A. L. Kornhauser. The interplay between fleet size, level-of-service and empty vehicle repositioning strategies in large-scale, shared-ride autonomous taxi mobility-on-demand scenarios. Technical report, 2017.

Appendices

:

A Expended description and discussion of used methods

A.1 Differential evolution

Since this method is used in the work, we present its idea in detail. We call the set of input arguments of the direct problem an "individual". A population is a vector of N such individuals (N is one of the parameters of the method). Each individual can be chosen either on the basis of some considerations, or randomly from the modelling space of the direct problem. Everything that will happen to the population will be referred to as evolution. The goal of evolution is to find an individual or individuals for whom the solution of the direct problem gives a minimal discrepancy. So, what evolutionary transformations will the population undergo?

- **Reproduction of descendants.** For an arbitrary vector T from the old population, a new virtual descendant V is created. This descendant has three parents and is calculated using the formula $V = X_i + F(X_j - X_k)$, where F is called the force of mutation and is also one of the parameters of the method. Each parent is chosen randomly from the set $1 \dots N$ and, of course, all of them (T, X_i, X_j, X_k) must be different.
- **Mutation.** The descendant of V obtained in the previous step can mutate with probability p. For each of the components $W_i = T_i$ with probability $1 - p$ and $W_i = V_i$ with probability p.
- **Selection.** Vectors T and W are passed through a sieve of natural selection. For this, the values of the discrepancies (objective functions) for T and W are compared and only one of them remains the strongest in the population. The method can lead to the fact that in a particular competition the strongest at the moment, but unpromising from the point of view of the future individual, wins. Increasing the algorithm parameter N smooths the negative sides of this fact.

A.2 Comparison of empty vehicle redistribution methods

In this subsection we provide the complete description of the evaluation of different redistribution algorithms on a simple linear network, considered in 4.4.3.

We consider a single line network with two stations and two vehicles (Figure -1). Each station has one client waiting for a vehicle. The driving time between the two stations is 5 minutes. We will now evaluate four cases with different locations of vehicles.

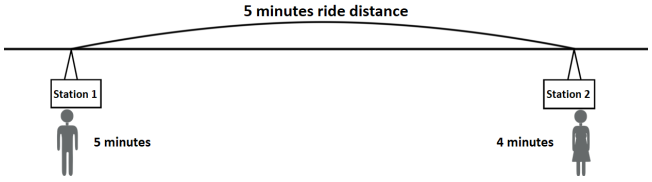


Figure -1 – The simple line network.

The first case of vehicle location is shown in the Figure -2. The first vehicle is located at 3 minutes ride time from station 1 and the second vehicle is located between station 1 and station 2, with a ride time of 1 minute to station 1 and 4 minutes to station 2.

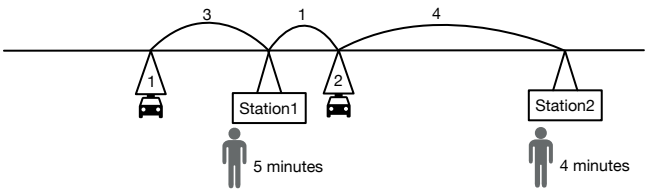


Figure -2 – The sample line network. Case 1.

The **SNN** algorithm will in each optimisation step serve the longest waiting client first with the nearest empty vehicle. The longest waiting client is in station 1, and the nearest vehicle is vehicle 2, which will thus be assigned to the client in station 1. Then there is one waiting client in station 2 that has not yet been assigned. To this client the remaining unassigned empty vehicle 1 will be assigned.

Therefore, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 6 minutes (5 min + 1 min drive time).
- For the client in station 2: 12 minutes (4 min + 8 min drive time).

Thus for **SNN** the maximum waiting time would be 12 min and the average waiting time 9 min.

The **STN** algorithm will for each optimisation step evaluate all the vehicle-station travel times (in drive time) and assign the nearest vehicle-station pair. In this case,

- the V1-St1 travel time is 3 minutes,
- the V1-St2 travel time is 8 minutes,
- the V2-St1 travel time is 1 minute,
- the V2-St2 travel time is 4 minutes.

Thus, the algorithm will assign vehicle 2 to the station 1.

In the next optimisation step, the algorithm will assign the remaining vehicle 1 to the remaining client in station 2.

The dispatching scheme in this case is therefore the same as for SNN. Thus, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 6 minutes (5 min + 1 min drive time).
- For the client in station 2: 12 minutes (4 min + 8 min drive time).

Thus for **STN** as in **SNN** the maximum waiting time would be 12 min and the average waiting time 9 min.

The **IBR** algorithm will sort all the stations by their index 4.4.1, and assign the nearest vehicle to the station with the highest index.

The index for station 1 is the (dis-)utility of the longest waiting client in this station including the time it takes for the nearest vehicle to get to the station. It will take 1 minute for the nearest vehicle to get to station 1 (vehicle 2), thus the index will be equal $u(5+1) = u(6)$. The second station index is $u(4+4) = u(8)$.

The utility function is monotonously increasing thus $u(8) > u(6)$. Based on the algorithm, the station with maximal index (station 2) will be served with the nearest vehicle (vehicle 2).

In the next optimisation step, the station 1 will be served with vehicle 1.

Thus, the waiting times for the clients, at time of departure will be:

- For the client in the station 1: 8 minutes (5 min + 3 min drive time).
- For the client in the station 2: 8 minutes (4 min + 4 min drive time).

Thus for **IBR** the maximum waiting time would be 8 min and the average waiting time 8 min.

Therfore, in case 1 the **IBR** method shows the best results for both maximal and average client waiting times.

The second case of vehicle location is shown in the Figure -3. The second vehicle is located at 7 min ride time from station 1 and the first vehicle is located at 1 min ride time from station 1.

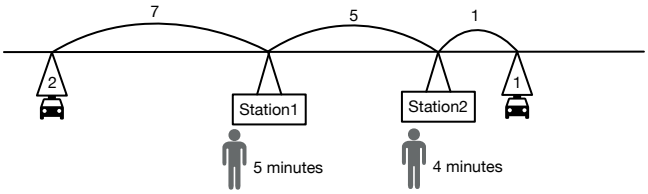


Figure -3 – The sample line network. Case 2.

The SNN algorithm will assign the vehicle 1 to the station 1, and the vehicle 2 to the station 2.

Therefore, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 11 minutes (5 min + 6 min drive time).
- For the client in station 2: 16 minutes (4 min + 12 min drive time).

Thus for SNN the maximum waiting time would be 16 minutes and the average waiting time 13.5 minutes.

The STN algorithm will assign vehicle 1 to the station 2 and the vehicle 2 to the station 1. Thus, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 12 minutes (5 min + 7 min drive time).
- For the client in station 2: 5 minutes (4 min + 1 min drive time).

Thus for STN the maximum waiting time would be 12 minutes and the average waiting time 8.5 minutes.

The IBR algorithm will send the vehicle 1 to the station 1 and the vehicle 2 to the station 2. The dispatching scheme in this case is therefore the same as for SNN. Thus, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 11 minutes (5 min + 6 min drive time).
- For the client in station 2: 16 minutes (4 min + 12 min drive time).

Thus for SNN and IBR the maximum waiting time would be 16 minutes and the average waiting time 13.5 minutes. Therefore, in case 2 the STN method shows the best results both in maximal and average client waiting times.

The third case of vehicle location is shown in the Figure -4. The first vehicle is located between station 1 and station 2, with a ride time of 1 min to station 1 and 4 min to station 2. The second vehicle is located at 5 min ride time from station 2.

The SNN algorithm will assign vehicle 1 to the station 1 and vehicle 2 to the station 2. Therefore, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 6 minutes (5 min + 1 min drive time).
- For the client in station 2: 9 minutes (4 min + 5 min drive time).

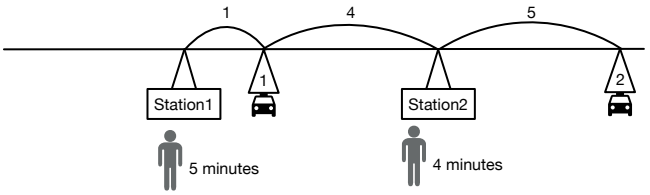


Figure -4 – The sample line network. Case 3.

Thus for SNN the maximum waiting time would be 9 minutes and the average waiting time 7.5 minutes.

The STN algorithm will assign vehicle 1 to the station 1 and vehicle 2 to the station 2. The waiting times for the clients at time of departure will be:

- For the client in station 1: 6 minutes (5 min + 1 min drive time)
- For the client in station 2: 9 minutes (4 min + 5 min drive time).

Thus for SNN and STN the maximum waiting time would be 9 minutes and the average waiting time 7.5 minutes.

The IBR algorithm will assign vehicle 1 to the station 2 and the vehicle 2 to the station 1. Thus, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 15 minutes (5 min + 10 min drive time).
- For the client in station 2: 8 minutes (4 min + 4 min drive time).

Thus for IBR the maximum waiting time would be 15 minutes and the average waiting time 11.5 minutes.

So, in case 3 the STN and SNN methods shows best results both in maximal and average client waiting times.

The fourth case of vehicle location is shown in the Figure -5. The first vehicle is located between station 1 and station 2, with a ride time of 4 min to station 1 and 1 min to station 2. The second vehicle is located at 2 min ride time from station 2.

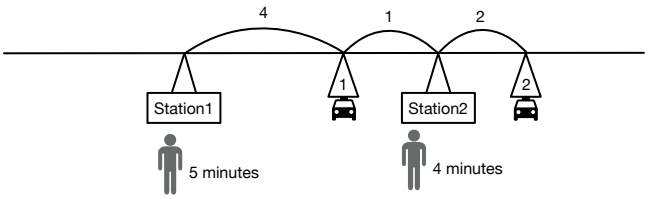


Figure -5 – The sample line network. Case 4.

The SNN algorithm will assign vehicle 1 to the station 1 and vehicle 2 to the station 2. Therefore, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 6 minutes (5 min + 4 min drive time).
- For the client in station 2: 9 minutes (4 min + 2 min drive time).

Thus for SNN the maximum waiting time would be 9 min and the average waiting time 7.5 min.

The STN algorithm will assign vehicle 1 to the station 2 and the vehicle 2 to the station 1. Thus, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 12 minutes (5 min + 7 min drive time).
- For the client in station 2: 5 minutes (4 min + 1 min drive time).

Thus for STN the maximum waiting time would be 12 minutes and the average waiting time 8.5 minutes.

The IBR algorithm will assign vehicle 1 to the station 1 and vehicle 2 to the station 2. The dispatching scheme in this case is therefore the same as for SNN. Thus, the waiting times for the clients, at time of departure will be:

- For the client in station 1: 6 minutes (5 min + 4 min drive time).
- For the client in station 2: 9 minutes (4 min + 2 min drive time).

Thus for SNN and IBR the maximum waiting time would be 9 minutes and the average waiting time 7.5 minutes.

So, in case 4 the IBR and SNN methods shows best results both in maximal and average client waiting times.

Summary

All the results obtained above are discussed in Subsection 4.4.3.

B OSM and GPS data analysis and processing

In this section we provide analysis of GPS and OSM data.

B.1 Kaggle Porto GPS data set

To analyse the existence of patterns in GPS data, we consider the data set of the city of Porto, Portugal, which was the training set of the Kaggle [139] competition ECML/PKDD 15: Taxi Trajectory Prediction (I) [143]. This data bundle provided our research with a data set containing trip information on the trajectories for all the 442 taxis running in the city of Porto from 01/07/2013 to 30/06/2014. This data set contains over 1.7 million data points.

These taxis operate through a taxi dispatch central, using mobile data terminals installed in the vehicles. Each data sample in the Porto data set corresponds to one completed trip. It contains a total of 9 features, described as follows:

$Trip_{id}$ contains an unique identifier for each trip.

$Call_{type}$ identifies the way used to demand this service. It may contain one of three possible values: central-based "A", stand-based "B" or otherwise "C".

$Origin_{call}$ contains an unique identifier for each phone number which was used to demand, at least, one service. It identifies the trip's customer if $Call_{type}$='A'. Otherwise, it assumes a NULL value.

$Origin_{stand}$ contains an unique identifier for the taxi stand. It identifies the starting point of the trip if $Call_{type}$='B'. Otherwise, it assumes a NULL value;

$Taxi_{id}$ contains an unique identifier for the taxi driver that performed each trip;

$Timestamp$ is Unix Timestamp (in seconds). It identifies the trip's start;

$Daytype$: It should identify the daytype of the trip's start. In the dataset description, it assumes one of three possible values (whereas the provided data are invalid):

'B' if the trip started on a holiday;

'C' if the trip started on a day before a type-B day;

'A' otherwise (i.e. a normal day, workday or weekend).

$Missingdata$ is FALSE when the GPS data stream is complete and TRUE whenever at least one location is missing.

$Polyline$ contains a list of GPS coordinates (i.e. WGS84 format) identified as [longitude, latitude] for each 15 seconds of trip. The last list item corresponds to the trip's destination while the first one represents its start.

In addition to the information revealed in the Section 6.2, we can mention the following.

Number of calls depending on time and day type

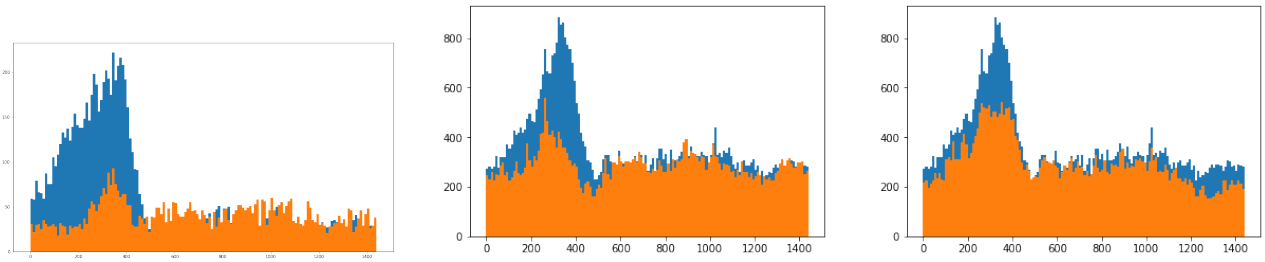


Figure -6 – The comparison of number of taxi called per 5-minute periods by call type C (a) in May, Saturday (blue) and Tuesday (yellow); (b) in May(blue) and July(yellow) for all weekdays; (c) in May (blue) and February (yellow) for all weekdays.

The histograms of calls count by call time reveal more detail about changes in the picture from month to month. A stronger night peak is sometimes noticeable for the type C calls in specific months. For example, in Figure -6(b), (type C calls in May) there is a night peak hour for Saturdays, strong in comparison with Tuesdays of the same month. Such a feature could be explained by the end of university year taking place in May in Porto.

Speed map

To obtain the basic road structure, the point appearance map with every placements of every vehicle in every moment was obtained in Figure -7.

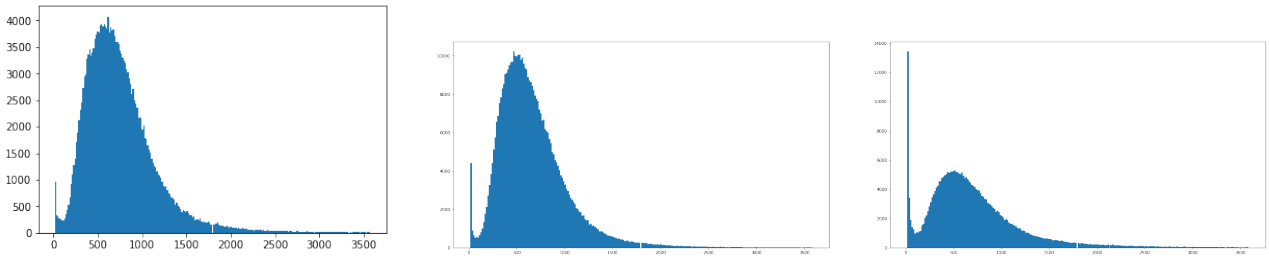


Figure -9 – The trip times (seconds) redistribution (a) call type A; (b) call type B; (c) call type C.

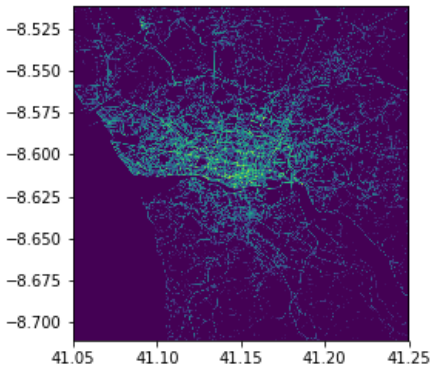


Figure -7 – The total point appearance map in Porto.

The speed map is an easy demonstration of a potential graph structure and it shows clearly the speed patterns on streets. For example, the maps in Figures -8a) and b) eliminate the fact that the speed in the central arteries is less subject of speed fluctuations caused by congestion.

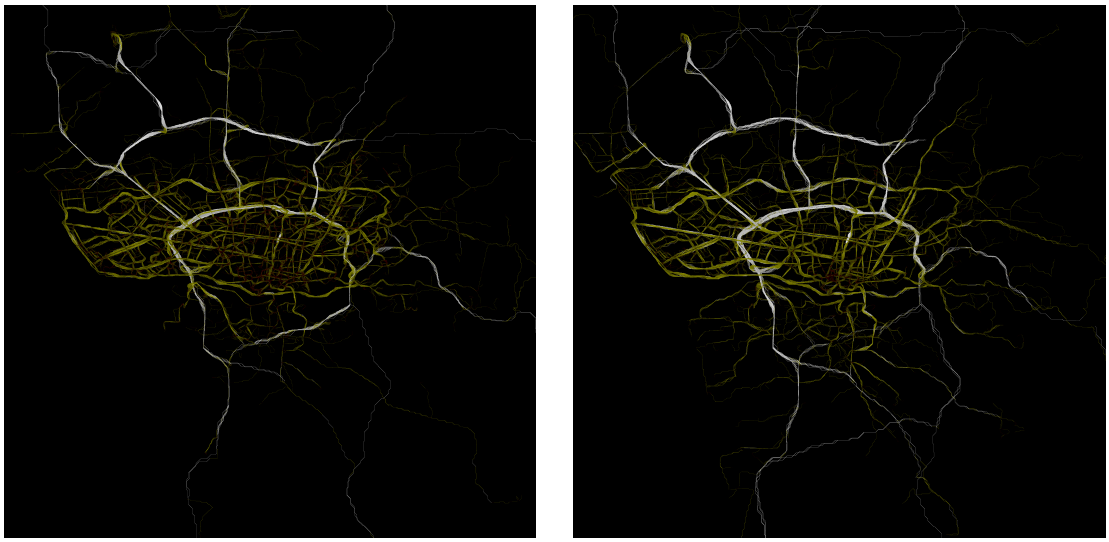


Figure -8 – The speed maps for Tuesdays a) from 7 to 10 am b) from 0 to 6 am.

Trip times in data set

We consider here all the trips made with travel times less than 1 hour (when considering longer times as a wrong data).

The trip time distributions are shown in Fig -9.

The corresponding expected travel times for different call times thus are the following:

For type A — 753 seconds.

For type B — 712 seconds.
For type C — 744 seconds.

The peaks for the trips with travel times equal to zero illustrate the cancelled calls. The histograms in Fig -9 show, that the probability of trip cancelling is significantly bigger for the calls of type C (made from street).

B.2 OSM data

OSM data description

The OSM data divide all routes (we show only roads with vehicle or bus access) by the following categories:

Roads: *motorway* is highway and has par default maximal speed 130 kmph and 2 lines;
trunk is a speedway and has characteristics of autoroute. It has par default maximal speed 110 kmph and 2 lines;
primary is a main road connecting big cities. In France it is, for example, national roads, big department roads or main arteries in the city. Par default it has maximal speed 50 kmph inside agglomeration, 90 kmph outside agglomeration and it has 2 lines;
secondary is a road that is part of the national network but isn't primary one. In France it is mostly department roads or important roads in the city. It has par default 2 lines and maximal speed is 50 kmph inside agglomeration, 90 kmph outside agglomeration;
tertiary is the road connecting villages or quarters inside cities. In France it can be considered as small roads or streets with transit traffic below a "secondary". It has par default 2 lines and maximal speed 50 kmph inside agglomeration, 90 kmph outside agglomeration;
unclassified is local service road, without transit traffic, or road in residential areas (e.g. commercial, industrial, hospital, small country road, etc.);
residential is service road or road within a residential area that does not have a specific nomenclature;
service is a road of access to building, industrial place, beach, camping etc or road inside big parking etc.

Linking roads: *motorway_link* is brace to access or exit a motorway;
trunk_link is brace to access or exit a trunk;
primary_link is brace to access or exit a "primary";
secondary_link is brace to access or exit a "secondary";
tertiary_link is brace to access or exit a "tertiary".

Special roads: *living_street* is a meeting area, sometimes called "pedestrian zone". It is road in residential area where pedestrians have absolute priority and are not obliged to move on sidewalks;
pedestrian is a pedestrian street. Street reserved mainly or exclusively for pedestrians in town, in commercial or residential area. Access of motorized vehicles is only permitted for very limited periods of the day;
track is road in principle without pavement (rural road, agricultural road, forest road, etc);
bus_guideway is a road reserved for guided buses (by rail, sidewalks, optical systems or radio);
escape is an emergency lane located on the side in long descents so that trucks and other vehicles can stop safely after breaking brakes;
raceway is a track of automobile/motorbike circuit;
road is a road or lane based on satellite imagery or cadastre, whose classification is unknown.

Paths: *footway* is the road reserved for pedestrians;
bridleway is a path for riders;
steps is stairs;
path is non-passable, non-specific or shared road for various uses.

Thus, for the purposes of this work, the roads with types "motorway", "trunk", "primary", "secondary", "tertiary", "unclassified", "residential", "service", "motorway_link", "trunk_link", "primary_link", "secondary_link", "tertiary_link", "living_street" and "path" were extracted.

OsmToRoadGraph output

The output format of the data using OsmToRoadGraph project is proposed in the following format:

```
<HEADER LINES STARTING WITH A DASH(#) CHARACTER>
<number of nodes>
<number of edges>
<id> <lat> <lon>
...
<s> <t> <length> <street_type> <max_speed> <forward> <backward>
...
```

where `<id> <lat> <lon>` sets vertices of the graph, and `<s> <t> <length> <street_type> <max_speed> <forward> <backward>` are arc parameters. `<forward> <backward>` are parameters that show if the road is one-way or two-way and the direction in one-way case. In fact, they are always (True,False) or (True,True), never (False,True) or (False, False), because arcs of only backward sense don't exist in OSM data being represented as arcs in the other direction.

B.3 Mixed data usage

GPS polyline fitting on the graph

The method described in Subsubsection 6.2.1 can be formalised the following way:

1. For each point of the polyline, take the N nearest points on the graph ("checkpoints").
2. Add arcs from the first point of the polyline to the nearest points of this point.
3. Add arcs from the nearest points of the last point to this point.
4. Let START be the first point of the polyline.
5. Let END be the last point of the polyline.
6. Add END at the end of the checkpoints list.
7. Start fitting the polyline to graph:
    - for every consecutive pair of checkpoints:
        - for each point of these checkpoints:
            - use Dijkstra algorithm from START to this point to get best paths.
    - take the common part of these paths taken by Dijkstra algorithm.
    - consider these points as the "good points".
8. Return the total path.

Speed calculation and clustering

The method described in Subsubsection 6.2.1 can be formalised the following way:

- Project every point of the polyline to the nearest arc of the predicted path.
- Remaining_arcs is the list of the predicted arcs.
- For every pair of projected points:
    - $e1$ and $e2$ are the arcs on which the points are projected.
    - interval_arcs are the arcs from remaining_arcs between $e1$ and $e2$;
    - if they don't exist anymore in remaining_arcs, abort;
    - remove these arcs except the last one from remaining_arcs;
- For every interval_arcs:
    - compute the speed between the 2 projected points.

C Scenarii evaluation

C.1 Ring network. Additional results

In case of full information, each client in the system can theoretically be served within its time windows. This can be done as follows:

- At station A, vehicle takes client a_n and moves to the destination of station B.
- At station B, vehicle takes client b_n and moves to the destination of station C.
- At station C, vehicle disembarks client a_n, takes client c_n and waits for 30 seconds of client d_n.
- Vehicle takes client d_n and moves towards station D.
- At station D, vehicle disembarks client b_n, takes client e_n and moves towards station A.
- At station A, vehicle disembarks client d_n and moves towards station A.
- At station B, vehicle disembarks clients c_n and e and waits for 30 seconds for client f_n.
- Vehicle takes client f_n and moves towards station A.
- Vehicle arrives to the station A, disembarks client f_n and goes to charge for 2 minutes.

A small charge of 2 minutes will fully restore vehicle's battery (based on provided vehicle characteristics). The cycle can be repeated the necessary number of times.

C.2 Grid network. Additional results

Table 1 shows results obtained from 100 runs of 400 iterations of reinforcement learning. It shows the main characteristics as dependencies of the chosen algorithm and of fleet-size. The column names in this table use the following abbreviations:

SBMH: Station-based reinforcement learning with heuristics (both ride-sharing and EVR), the average results for the 400th iteration;

CSBMH: Centralised station-based reinforcement learning with heuristics (both ride-sharing and EVR), the averaged result for 400th iteration.

The results given by centralised learning are better than the results given by non-centralised learning, but at the expense of program execution time. The more vehicles are in the system the more is time difference between the centralised and the non-centralised algorithm. This is due to the extra time needed for cross-matching over vehicles. The obtained vehicle loads show that under small fleet-size empty vehicles do not redistribute. However, the sharing rate is growing up in the beginning of fleet-size variation. This can be explained by the fact that clients who can be served by one vehicle can be located in different stations, and the vehicle position matters.

Nevertheless, the further increase in fleet-size provokes the increase in empty runs, which influences negatively both energy consumption and average load.

Table 1 – Different available vehicles modelling comparison

Method	5 vehicles		9 vehicles		18 vehicles		36 vehicles	
	SBMH	*CSBMH*	*SBMH*	*CSBMH*	*SBMH*	*CSBMH*	*SBMH*	*CSBMH*
Not served(clients)	50.1	47.5	35.2	24.8	25.5	14.2	4.7	2.8
Avg wait (min)	8.19	7.58	3.50	3.64	3.45	3.15	1.56	1.53
Max wait (min)	28	28	28	24	24	20	10	10
Total run (km)	73.4	80.6	107.5	117.6	119.4	126.1	231.4	228.3
Empty run (km)	0	0	0	0	20.6	35.8	71.5	60.7
TR_1 *(km)*	45.7	50.7	45.7	44.4	35.7	41.7	70.6	79.7
TR_2 *(km)*	23.8	21.8	37.7	51.5	31.8	23.8	69.9	65.7
TR_3 *(km)*	1.9	3.9	17.4	13.9	23.7	16.3	13.7	18.2
TR_4 *(km)*	2.0	4.2	6.7	7.8	7.6	7.9	5.9	4.0
Avg load (clients)	1.46	1.52	1.85	1.87	1.68	1.35	1.19	1.23
Exec time (sec)	0.33	10.6	0.41	15.8	0.43	37.6	0.59	81.0

C.3 Saclay network. Additional results

Figure -10 shows the client queue divided by client demand for each station. These results were obtained foe rush hour demand and they show, that there are no strong asymmetries across stations for the client queue length as a percentage of the station demand, indicating that all the stations are almost uniformly served despite the distance between them, and despite large variations in the demand.

In Figure -11, the main comparison results are shown for the seven strategies applied to off-peak demand .

The results of the simulation on the Saclay network for fleet-size of 60 vehicles under morning rush hour demand with and without empty vehicle redistribution are shown in Figure -12. The simulation was performed using VIPSIM program and the simulation time was equal to 1 hour.

- **clientsServed**: Number of clients served during the simulation period;
- **clientsNotServed**: Number of clients not served during the simulation period;

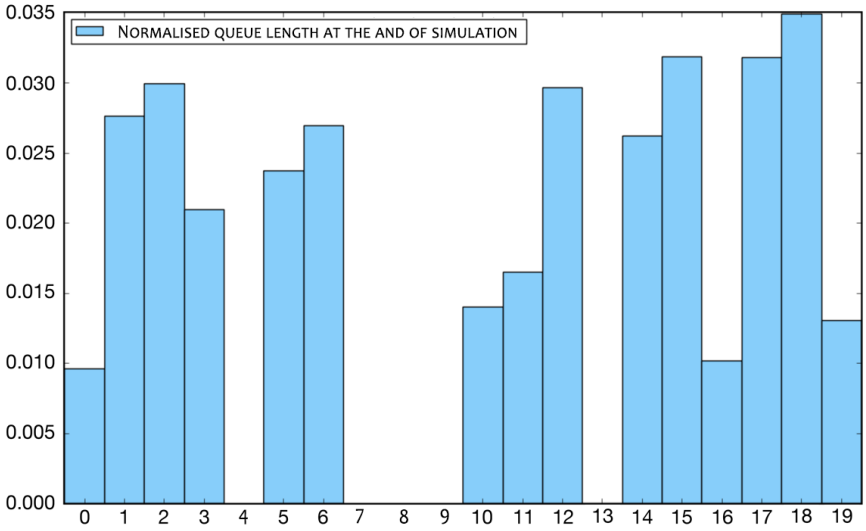


Figure -10 – Rush hour traffic: Client queue length at the end of the simulation divided by the station client demand for each station.

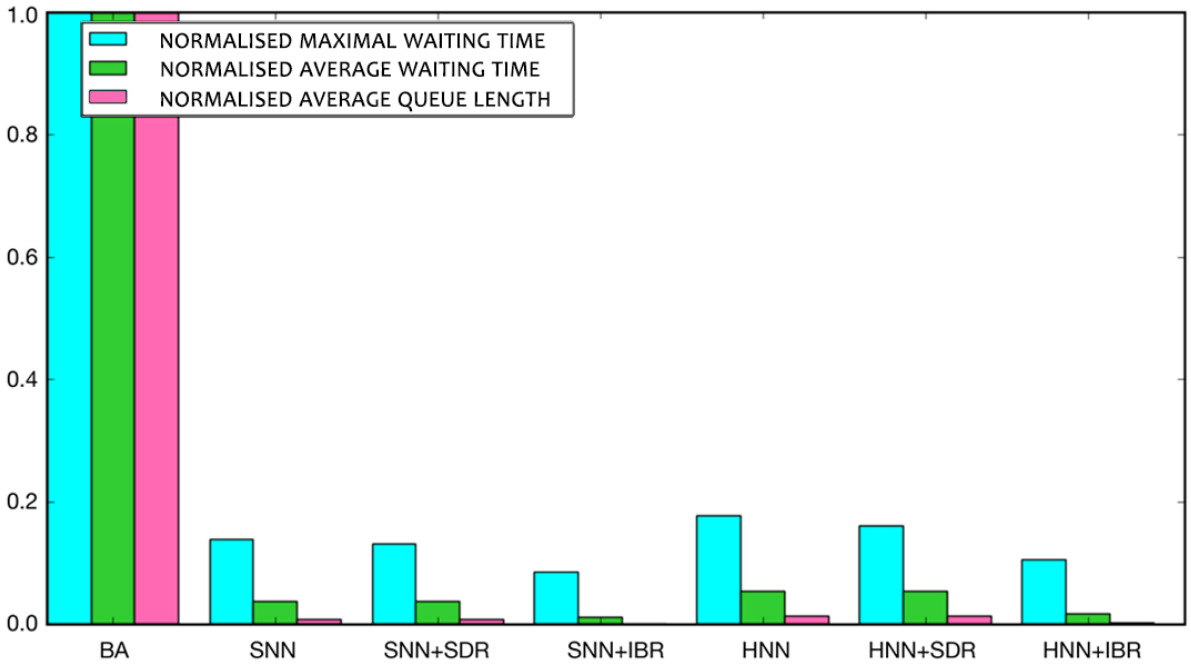


Figure -11 – Off-peak demand: comparison between different strategies.

- **maxClientsWait**: Maximal waiting time during the simulation (including not served yet clients), in seconds;
- **averageClientsWait**: Average waiting time during the simulation (including not served yet clients), in seconds;
- **vehicleKmLoaded**: Total run distance with clients, in kilometres;
- **vehicleKmEmpty**: Total run distance without clients, in kilometres;
- **vehicleLoadingRatio**: Average occupancy of vehicles during simulation time;
- **energyConsumptionEmpty**: Total energy consumption for runs without clients;
- **energyConsumptionLoaded**: Total energy consumption for runs with clients;
- **maxQueueLength**: Maximal number of waiting clients at one station;
- **averageQueueLength**: Average number of waiting clients at one station.

C.4 Stockholm network. Additional results

The strategies to compare between are:

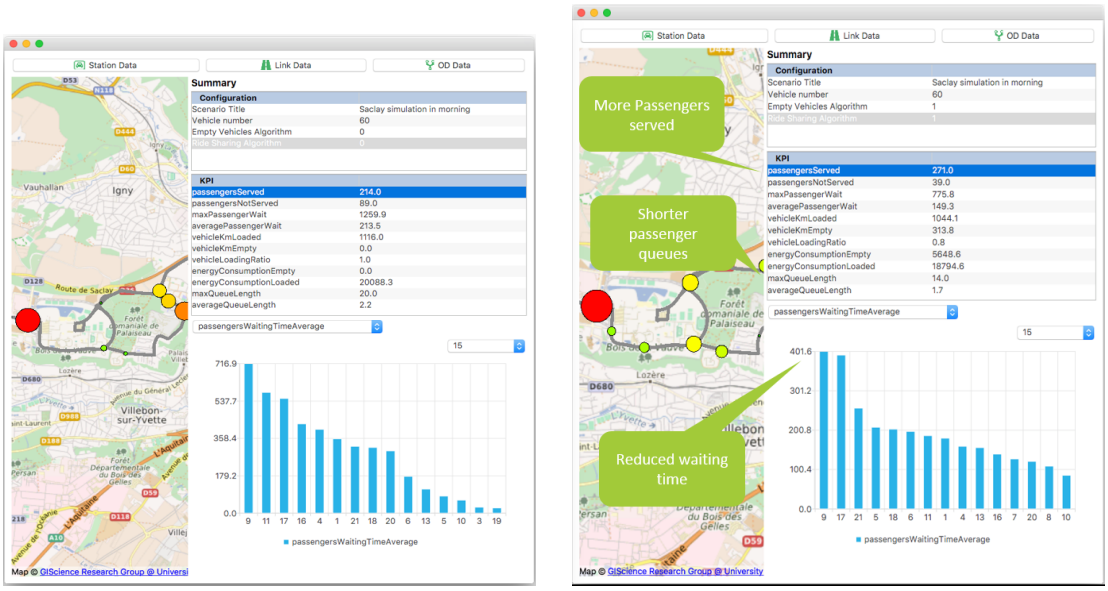


Figure -12 – The ride-sharing evaluation on Saclay under morning rush hour demand (a) Without ride-sharing; (b) With ride-sharing.

0.0 BA. Basic Allocation. This means no redistribution of empty vehicles between the stations, baseline for comparison with the other strategies
1.0 SNN. Simple Nearest Neighbours.
1.1 SNN+SDR. Simple Nearest Neighbours combined with Surplus/Deficit vehicle Redistribution
1.2 SNN+IBR. Simple Nearest Neighbours combined with Index Based Redistribution
1.3 SNN+IBRTL. Simple Nearest Neighbours combined with Index Based Redistribution Time Limited
2.0 HNN. Heuristic Nearest Neighbours
2.1 HNN+SDR. Heuristic Nearest Neighbours combined with Surplus/Deficit vehicle Redistribution
2.2 HNN+IBR. Heuristic Nearest Neighbours combined with Index Based Redistribution
2.3 HNN+IBRTL. Heuristic Nearest Neighbours combined with Index Based Redistribution Time Limited
3.0 HNNTL. Heuristic Nearest Neighbours Time Limited
3.1 HNNTL+SDR. Heuristic Nearest Neighbours Time Limited combined with Surplus/Deficit vehicle Redistribution
3.2 HNNTL+IBR. Heuristic Nearest Neighbours Time Limited combined with Index Based Redistribution
3.3 HNNTL+IBRTL. Heuristic Nearest Neighbours Time Limited combined with Index Based Redistribution Time Limited
4.0 STN. Send The Nearest
4.1 STN+SDR. Send The Nearest combined with Surplus/Deficit vehicle Redistribution
4.2 STN+IBR. Send The Nearest combined with Index Based Redistribution
4.3 STN+IBRTL. Send The Nearest combined with Index Based Redistribution Time Limited

In the tables below, we use the following abbreviations:

NotServ Number of not served clients,
AvW Average waiting time within served clients, in minutes,
MaxW Maximal waiting time within served clients, in minutes,
PrNotServ Percentage of not served clients,
Queue Queue length at the end of the simulation. The value of queue length for the simulation is the number of passengers not served yet at the end of simulation. All the passengers that are staying at the end of the simulation may be served later within their time windows.

The tables below are organised as follows:

The first column is the type of combination presented in the row. We investigate two cases here. First, clients with time window (columns 2-6) where clients not served within 10 minutes will quite the system. Second, clients are waiting as long as they are served and never leave simulation until they are served (last 3 columns). The «—» in the table mean that this data is not available, because uses TL versions of algorithms.

Table 2 – The comparison between different redistribution methods for Stockholm network. Full table.

	with time windows					without time windows		
Strategy	NotServ	AvW	MaxW	PrNotServ	Queue	Queue	AvW	MaxW
0.0 BA	1340	7.9	10	92.4	64.74	1426	28.6	84.9
1.0 SNN	140.88	6.4	10	9.73	38.58	76.5	8.8	24.7
1.1 SNN+SDR	129.44	6.2	10	8.9	35.22	115.6	11.4	27.7
1.2 SNN+IBR	118.36	6.2	10	8.1	31.76	48.9	9.1	26.5
1.3 SNN+IBRTL	80.14	5.9	9.99	5.5	20.9	—	—	—
2.0 HNN	289.88	6.0	9.99	19.9	38.56	112.4	11.9	29.5
2.1 HNN+SDR	238	5.8	9.99	16.4	30.78	111.6	11.5	28.9
2.2 HNN+IBR	224.98	5.8	9.99	15.5	23.76	124.0	13.4	33.4
2.3 HNN+IBRTL	112.24	6.2	9.99	7.7	18.8	—	—	—
3.0 HNNTL	120.68	6.2	9.98	8.3	33.86	—	—	—
3.1 HNNTL+SDR	110.12	6.2	9.99	7.6	40.28	—	—	—
3.2 HNNTL+IBR	85.92	6.1	9.99	5.9	25.08	—	—	—
3.3 HNNTL+IBRTL	97.98	6.0	9.98	6.7	20.64	—	—	—
4.0 STN	65.50	2.5	9.95	4.5	11.82	17.8	3.3	28.8
4.1 STN+SDR	59.82	2.4	9.91	4.1	11.84	16.9	3.1	27.5
4.2 STN+IBR	55.5	2.9	9.95	3.8	9.58	9.1	3.455	18.2
4.3 STN+IBRTL	37.42	2.7	9.96	2.6	8.14	—	—	—

Table 3 – The comparison between different redistribution methods for Stockholm network, small demand. Full table.

	with time windows					without time windows		
Strategy	NotServ	AvW	MaxW	PrNotServ	Queue	Queue	AvW	MaxW
0.0 BA	587.46	8.22	9.99	80.97	23.84	667	35.7	99.5
1.0 SNN	1.2	1.14	7.50	0.16	2.04	0.8	1.26	7.93
1.1 SNN+SDR	1.72	1.33	7.67	0.23	1.94	1.24	1.14	7.85
1.2 SNN+IBR	0.02	0.68	6.32	0.003	0.02	0.14	0.67	6.18
1.3 SNN+IBRTL	0.14	0.67	6.22	0.002	0.12	—	—	—
2.0 HNN	3.34	1.33	7.85	0.46	2.34	0.86	1.22	8.10
2.1 HNN+SDR	2.14	1.31	7.87	0.29	1.98	1.18	1.21	7.96
2.2 HNN+IBR	0.22	0.65	6.18	0.03	0.2	0.16	0.76	6.64
2.3 HNN+IBRTL	0.14	0.71	6.23	0.02	0.18	—	—	—
3.0 HNNTL	1.38	1.23	7.59	0.19	1.4	—	—	—
3.1 HNNTL+SDR	1.66	1.15	7.43	0.23	2.2	—	—	—
3.2 HNNTL+IBR	0.06	0.66	6.04	0.008	0.12	—	—	—
3.3 HNNTL+IBRTL	0.16	0.79	6.75	0.02	0.06	—	—	—
4.0 STN	0.7	0.67	7.90	0.1	0.34	0.34	0.79	8.73
4.1 STN+SDR	0.46	0.66	7.52	0.06	2.04	0.62	0.73	8.42
4.2 STN+IBR	0.06	0.40	6.43	0.008	0.18	0.2	0.45	6.01
4.3 STN+IBRTL	0.02	0.43	6.02	0.003	0.12	—	—	—

Table 4 – The comparison between different redistribution methods for Stockholm network with real demand data. Full table.

	with time windows					without time windows		
Strategy	NotServ	AvW	MaxW	PrNotServ	Queue	Queue	AvW	MaxW
0.0 BA	1252	8.55	10	86.3	146	1444	21.16	75.99
1.0 SNN	121	5.54	10	8.36	138	297	5.94	29.4
1.1 SNN+SDR	118	5.47	10	8.16	143	249	6.03	25.1
1.2 SNN+IBR	106	5.27	10	7.30	125	284	7.99	36.66
1.3 SNN+IBRTL	81	4.92	9.97	5.55	122	—	—	—
2.0 HNN	170	2.10	9.99	11.70	131	297	5.14	33.6
2.1 HNN+SDR	240	2.95	9.99	16.56	132	358	7.09	30.04
2.2 HNN+IBR	263	3.74	10	18.16	137	286	5.35	33.84
2.3 HNN+IBRTL	56	3.84	10	3.88	116	—	—	—
3.0 HNNTL	89	4.75	10	6.16	126	—	—	—
3.1 HNNTL+SDR	83	3.49	9.96	5.75	125	—	—	—
3.2 HNNTL+IBR	81	4.33	9.99	5.59	124	—	—	—
3.3 HNNTL+IBRTL	76	3.91	9.98	5.22	123	—	—	—
4.0 STN	42	2.05	9.87	2.86	63	82	2.33	32.80
4.1 STN+SDR	48	1.91	9.95	3.30	57	68	2.43	31.83
4.2 STN+IBR	39	1.98	9.93	2.69	75	86	2.97	21.40
4.3 STN+IBRTL	43	2.62	9.97	2.93	84	—	—	—

D Abbreviations

WHO World Health Organization
UN United Nations
GDP Gross Domestic Product
IoT Internet of Things
aTaxi autonomous Taxi
NP-hard Non Polynomial hard
OSM OpenStreetMap
GPS Global Positioning System
PRT Personal Rapid Transit
TAAS Transport as a service
MAAS Mobility as a Service
RP Revealed Preferences
SC Stated Choice
VOT Value of time
VTTC Value of travel time changes
RU Random Utility
RV Random Valuation
BEV Battery Electrical Vehicle
HEV Hybrid Electric Vehicle
pHEV Plug-in Hybrid-Electric Vehicle
ICE Internal combustion engine
EV Electric Vehicle
EVSE Electric vehicle service equipment
E-VRP Electric vehicle relocation problem
E-VRPTW EVRP with time windows and charging stations
EVR Empty vehicle redistribution
VRP Vehicle Routing Problem
TSP Travelling Salesman Problem
SCRAM Scalable collision-avoiding role assignment with minimal makespan
DVAPC Dynamic vehicle allocation problem for car-sharing
OD Origin-Destination
NN Nearest neighbour
SV heuristic Sampling and Voting heuristic
FCFS First-come, first-served
WG Wait, then go
FNVS Fixed number of vehicles in station
PDP Pick up and delivery problem
E-VRP-RS] Electrical vehicle routing problem with ride-sharing
MDVRP Vehicle routing problem with multiple depots
AV Autonomous vehicle
MATSim The multi-agent transport simulation
DVRP Dynamic VRP
SAV Shared autonomous vehicle
DRS Dynamic ride-sharing
HOV High Occupancy Vehicle
ADMM Alternate Direction Method of Multipliers
SA Simulated annealing
DARP-M Dial-a-Ride Problem with Money as an Incentive
GRASP Greedy randomized adaptive search procedure
FIPA Foundation for Intelligent Physical Agents
VMT Vehicle miles travelled
BOC Bayes optimal classifier
BMA Bayesian model averaging
MCMC Markov Chain Monte Carlo
BMC Bayesian model combination
GCRF Gaussian conditional random field
PPM Partial Matching
VOMM Variable-order Markov Model
VANETs Vehicular Ad Hoc Networks
P2P Peer-to-Peer
LDA Linear Discriminant Analysis
SVM Support vector machine

CAR model Conditional Auto-Regressive model
STCAR model Spatial Temporal Conditional Auto-Regressive model
GMCAR model Generalized Multivariate Conditional Auto-Regressive model
MSTAR model Multivariate spatial-temporal autoregressive model
ITL Inferential theory of learning
ANNs Artificial neural networks
NN Neural network
CFL Capacitated facility location
MIP Mathematical Integer Programming
DE Differential evolution
PCA Principal component analysis
ICA Independent Component Analysis
LDA Linear discriminant analysis
RL Reinforcement learning
MARL Multi-agent reinforcement learning
DRL Deep reinforcement learning
COMA Counterfactual multi-agent
IAC Independent actor-critic
QEN Q estimator network
TDNN Time-delay neural network
FIS Fuzzy inference system
TD Temporal-difference
VRP-TW-F Vehicle routing and scheduling problem with time window-forecasted
GA Genetic Algorithms
LA Linear Automata
LRI Linear Reward-Inaction
LRP Linear Reward Penalty
ASH-learning Afterstate H-learning
CBM Coalition-based metaheuristic
STN Send The Nearest
IBR Index-Based Redistribution
HNN Heuristic Nearest Neighbours
HNNTL Heuristic Nearest Neighbours Time Limited
IBRTL Index Based Redistribution Time Limited
FCM Fill Clustering Method
SCM Spill Clustering Method
BA Basic Allocation
SNN Simple Nearest Neighbours
SDR Surplus/Deficit vehicle Redistribution
CS Client surplus
TW Time Window
RJVI rectangular LAPJV
LAPJV Linear Assignment Problem solver using Jonker-Volgenant
ACVRP CVRP with asymmetric cost matrix
CVRP VRP with chargings
AP Assignment Problem
MINLP Mixed integer nonlinear programming
KDE Kernel Density Estimation
ECML PKDD European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases
ENSTA Ecole Nationale Supérieure de Techniques Avancées
ONERA French aerospace lab
SNCF
RER Réseau Express Régional (commuter trains)
BRT Bus Rapid Transit
STIF Ile de France Transport Authority
VISUM Traffic planning software by PTV Group
SOTW Searching on the way
SOTA Searching in the area of the origin and/or destination
VDMNH Vehicle-based model without heuristics
VDMH Vehicle-based model with heuristics
VIPSIM integrated passenger transport simulator
VEDECOM Institut du Véhicule Décarboné Communicant et sa Mobilité

E Notations

t modelling time

$G = G(V, A)$ directed graph of the city

$v \in V$ vertex of the city graph

$a \in A$ arc of the city graph

$init_a$ initial vertex of arc a

ter_a terminal vertex of arc a

$distance_a$ the distance of the arc a

$speed_a(t)$ the speed of crossing the arc a at the starting time t

$time_a(t)$ the time to cross the arc a at the starting time t

$T[v_1, v_2](t)$ minimal travel time between vertices v_1 and v_2 at the starting time t

$z \in Z$ zone in the system

V_z set of vertices of zone z

A_z set of arcs of zone z

$\gamma \in \Gamma$ stations in the system

v_γ position of the station γ

S_γ^{max} capacity of the station γ

$Type_\gamma$ type of the station γ

$p \in P$ client (passenger group, travelling together)

N_p number of passengers for client p

$v_p^o \in V$ origin of client p

$v_p^d \in V$ destination of client p

t_p^0 minimal accepted boarding time of the client p

t_p^1 maximal accepted boarding time of the client p

w_p^{max} maximal walking distance of client p

$TW_p = t_p^1 - t_p^0$ time window size of the client p

t_p^i ideal pick up time of the client p

$rtosh_p$ readiness to share of the client p

t_p^{app} the taxi call time of the client p

u_p^{min} utility constraint of the client p

$Ratio_{max}$. the maximal possible ride-sharing ratio delay

$\xi \in \Xi$ vehicle in the system

S_ξ capacity of the vehicle ξ

D_ξ^{max} maximal travel distance on one charge of the battery of the vehicle ξ

Ch_ξ charging time from empty to full battery of the vehicle ξ

$\Xi_z(t)$ set of vehicles inside zone z at moment t

$P_z(t)$ set of clients who have already booked a taxi in zone z

$IP_z(t)$ predicted intensity of new client arrivals in zone z for moment t

$S_\gamma(t)$ current occupancy of station γ at moment t

$P_\gamma^{available}(t)$ empty slot heuristical probability of station γ at moment t

r trip in the system

$v^o(r)$ origin vertex of trip r

$v^d(r)$ destination vertex of trip r

$P(r) \equiv r^P$ set of clients inside for trip r

$\xi(r) \equiv r_\xi$ vehicle carrying out the trip r

$tend(r)$ time to finish the trip r

R^a client a trip sequence (trajectory)

R_ξ vehicle ξ trip sequence (trajectory)

$D_\xi(t)$ current charging of vehicle ξ

t_p^{dep} real departure time of client p

w_p^{real} real walking distance of client p

w_p^{prop} proposed walking distance for client p

t_p^{prop} proposed taxi arrival time for client p

$u_{waiting}(t_p^{prop}, t_p^i, t_p^0, t_p^1)$ proposed waiting utility of the client p

$u_{walking}(w_p, w_p^{prop})$ proposed walking utility of the client p

$u_{rsh}(rtosh_p)$ proposed ride-sharing utility of the client p

$u_{proposedtrip}(p, R)$ proposed ride R utility of the client p

$u_{waiting}^R(t_p^{real}, t_p^{prop}, t_p^0, t_p^1)$ utility function of real waiting time of the client p

$D(R^p)$ delay utility function for the client's p sequence of trips

TTT_p Total Trips Time of the client p, thus, the total time from the boarding to the disembarking of the client p.

$MPDT_p$ minimal possible driving time for the client p

L_r length of the trip r

F_p is flag of the serving of the client p within its time window constraints

$T_\gamma^{available}(t)$ waiting time of the empty slot at station γ

$u_p(t_p^{dep} - t_p^0)$ utility of client p for IBR method

$P_{s,arr>0}(t_1, t_1 + \tau)$ probability of at least one client arrival to the station s during the time period $(t_1, t_1 + \tau)$

$\xi_s^{nearest}$ nearest available vehicle to the station $s \in V$

CS_s client surplus on the station s

I_s index of the station s

T_s maximal waiting time among the clients at the station s

$T_{s,X}$ the time expectancy of X clients arriving to the station s

T_s^m maximal waiting time within client at the station s at the moment of the nearest vehicle arriving to the station s

$t_{m-1,1-\alpha/2}$ critical value of the two-tailed t-distribution at a level α of significance, given $m-1$ degrees of freedom

ZS_z vehicle surplus in zone z

I_z index of zone z

$|\Xi_z^{avail}|$ number of available vehicles in zone z

$|P_z^{demand}|$. number of clients that are not served yet and have no one vehicle assigned in zone z

ZS_z zone z surplus

T_z sum of waiting times among clients of zone z

$SUM_z(t_1, t_2)$ predicted sum of waiting times at the moment of vehicle arrival t_2 to the zone

I_z index of zone

S_i^ξ rating of client orders near vertex i

A_i^ξ rating of client arrivals to vertex i

D_i^ξ rating of client destinations to vertex i

W_i^ξ rating of time window of clients in vertex i

N_i^ξ number of requests near vertex i

Np_i^z number of positive results from vertex i

B_i^ξ predicted time window size in vertex i

List of Figures

List of Tables

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Machine Learning pour la gestion distribuée et dynamique d'une flotte de taxis et navettes autonomes

Mots clés : Véhicules autonomes, Gestion des taxis, Apprentissage par renforcement

Résumé : Dans cette thèse sont étudiées des méthodes de gestion des systèmes urbains de taxis autonomes électriques partagés. Nous considérons un contexte en ligne dans lequel les demandes des clients se produisent au fil du temps, et où les véhicules sont disponibles pour le co-voiturage et nécessitent une gestion de la recharge électrique. Nous proposons des heuristiques basées sur la décomposition de ce problème qui inclut la répartition du réseau routier et la mise en évidence de sous-problèmes tels que la gestion de la charge, la redistribution des véhicules vides et le partage de trajet dynamique.
L'une des contributions principales de la thèse est le développement de méthodes heuristiques pour les problèmes d'optimisation locaux. Nous proposons un ensemble de nouvelles méthodes de redistribution des véhicules vides, certaines sont proactives, prenant en compte à la fois la demande actuelle et la demande future anticipée, et certaines autres réactives, qui agissent uniquement sur la demande actuelle. Une autre contribution principale est l'approche d'apprentissage par renforcement à différents niveaux, en fonction de la granularité du système : un modèle RL basé sur les stations pour les petits réseaux, et un modèle RL basé sur les zones pour les plus grands réseaux et où les agents sont des zones de la ville obtenus par partitionnement.
L'évaluation des performances des méthodes proposées est réalisée sur un ensemble de réseaux routiers de nature et de taille différentes. Ces méthodes donnent des résultats prometteurs qui surpassent les autres méthodes testées et les données réelles sur les performances du système de taxi, en termes de nombre de passagers satisfaits lorsque la taille de la flotte est fixe.

Title: Machine Learning for the distributed and dynamic management of a fleet of autonomous taxis and shuttles

Keywords: Autonomous vehicles, Taxi management, Reinforcement learning

Abstract: In this thesis are investigated methods to manage shared electric autonomous taxi urban systems. We consider an online context in which customer demands occur over time, and where vehicles are available for ride-sharing and require electric recharging management. We propose heuristics based on the problem decomposition, which include road network repartition and highlighting of sub-problems such as charging management, empty vehicle redistribution and dynamic ride-sharing.
One of the main contributions of the thesis is the development of heuristic methods for local optimisation problems. A set of new methods for empty vehicle redistribution is proposed, some are proactive, that is taking into account both current demand and anticipated future demand, while others are reactive methods, which act based on current demand only. Another main contribution is the reinforcement learning approach at different levels depending on the targeted granularity of the system: station-based RL model for small networks and zone-based RL model, where the agents are zones of the city obtained by partitioning, for huge ones. The set of proposed solutions is designed for use in various transport networks.
A performance evaluation of the proposed methods is provided for a set of road networks of different natures and sizes. These methods provide promising results outperforming the other tested methods and the real data of the taxi system performances, in terms of the number of satisfied passengers under fixed fleet size.

Maison du doctorat de l'Université Paris-Saclay
2ème étage aile ouest, Ecole normale supérieure Paris-Saclay
4 avenue des Sciences,
91190 Gif sur Yvette, France