Politecnico di Torino

Master of Science in Electronic Engineerig

Academic Year 2020/2021

October 2021

How to control electric autonomous taxi fleets in an energy efficient way

Optimal control of Electric Autonomous Mobility On Demand transportation systems in a predictive framework with smart charging

Supervisor:
Alessandro Rizzo
Politecnico di Torino

External Supervisors:
Balázs Adam Kulcsár
Chalmers University of Technology
Sten Elling Tingstad Jacobsen
Volvo Cars

Candidate:
Edoardo Ascenzi

Master's thesis 2021

How to control electric autonomous taxi fleets in an energy efficient way

Optimal control of Electric Autonomous Mobility On Demand transportation systems in a predictive framework with smart charging

Edoardo Ascenzi

Department of Electronics and Telecommunications
Electronic Engineering for Industrial Applications
Politecnico di Torino

Department of Electrical Engineering
Division of System and Control
Chalmers University of Technology

Department of Vehicle Energy Efficiency
Data Analysis and Strategy Team
Volvo Cars

Turin, Italy 2021

How to control electric autonomous taxi fleets in an energy efficient way
Optimal control of Electric Autonomous Mobility On Demand transportation systems in a predictive framework with smart charging

Supervisor: Alessandro Rizzo, Politecnico di Torino
External Supervisor: Balázs Adam Kulcsár, Chalmers University of Technology
External Supervisor: Sten Elling Tingstad Jacobsen, Volvo Cars
Manager: Jonas Henningsson, Volvo Cars

Master's Thesis 2021
Department of Electronics and Telecommunications
Master Degree in Electronic Engineering for Industrial Applications
Politecnico di Torino
Corso Castelfidardo, 10138 Turin, Italy

Typeset in LaTeX
Turin, Italy 2021

HOW TO CONTROL ELECTRIC AUTONOMOUS TAXI FLEETS IN AN ENERGY EFFICIENT WAY
Optimal control of AMoD electric transportation systems in a predictive framework with smart charging

Edoardo Ascenzi
Politecnico di Torino - Department of Electronics and Telecommunications
Chalmers University of Technology - Department of Electrical Engineering
Volvo Cars - Vehicle Energy Efficiency

Preface

This master thesis project was carried out entirely in Gothenburg, Sweden, at Volvo Cars. The team that developed the work was composed of me and Francesca Palanza, a peer student of Mechanical Engineering at Politecnico di Milano. The study of the literature and the elaboration of the used algorithm were performed together. My teammate focused more on the implementation of the Sample Network (Section 3.1), briefly described in this report. At the same time, I worked more in the Real Network implementation described in Section 3.3. In the end, we drew the conclusion of the project together by analyzing the set of results acquired.

Abstract

How can the development of new technologies impact the future of personal urban mobility? What direction can we give to the advancements in fields such as automation and electric vehicles, so that they can become our allies in the development of a more sustainable world? This thesis tries to provide an answer to those questions by investigating the emergent world of Autonomous Mobility on Demand (AMoD). Such new technology promises to revolution our cities and the way we travel across them, by releasing fleets of autonomous electric taxis that are able to pick up passengers in few minutes upon request and carry them to wherever they might ask. Many are the market players as well as the academic researchers currently taking part in this new technological challenge, and if some of its aspects have already been addressed and studied in detail, there are still numerous open issues that need to be investigated. In order to make this technology interesting enough to be adopted, so that it can cooperate with the already existing forms of transport and hopefully, one day, even replace some of them, it is necessary to enhance it and optimize its functioning. In this research, a new algorithm for an efficient control of such fleets of autonomous electric taxis is developed and tested. It includes the optimal assignment of passengers to vehicles, a periodic redistribution of free taxis along the network operated in a predictive framework, and a smart handling of vehicles' charging. The policy developed here shows good results in terms of a reduction of mean customers' waiting time and distance travelled by the taxis when not transporting passengers, resulting in an increased quality of the service and in a reduced energy consumption of the overall system. Some of the questions that are raised on the subject still remain open after this research, thus constituting interesting starting points for possible future studies.

Keywords: Traffic modelling and simulation, model predictive control, assignment problem, rebalancing, sustainability, efficiency optimization, charging optimization

Acknowledgment

First of all, I would like to thank my supervisor from Politecnico di Torino, Alessandro Rizzo for accepting my thesis proposal and helping me throughout all the steps necessary to complete the journey of my Master.

A big thanks goes to my supervisor from Volvo Cars, Sten Elling, for giving me the opportunity to discover and learn this topic which I found really interesting. I wish him the best of luck for the continuation of his PhD project.

I would also like to thank Jonas and the VEE group at Volvo Cars for welcoming me and making me feel a part of the team. In particular, I am grateful for the support and valuable advices that Björn has provided me along the way.

A special thanks goes to Balazs, my supervisor from Chalmers university. He accompanied and guided me through the hard but very rewarding journey of my master thesis research, motivating me more and more everyday. He did not only teach me important lessons in engineering but also inspire me with his incredible kindness and brightness. He always showed me unconditioned support, and this helped me to believe in myself and in the project. Working with him was a real privilege and a pleasure.

Edoardo Ascenzi, Turin, October 2021

Contents

List of Figures

List of Tables

Acronyms

AMoD Autonomous Mobility on Demand. vi, 3–5, 35, 37, 59–62
AMoDeus Autonomous Mobility on Demand Simulator. xii, 25, 46–48, 50–52, 62
AR Autoregressive. 6, 7, 10, 16, 55, 59, 62

CT Customers per Taxi. xi, 35–38, 56

MATSim Multi-Agent Transport Simulation. 46
MILP Mixed Integer Linear Programming. 4, 43
MINLP Mixed Integer Non Linear Programming. 43
MIQP Mixed Integer Quadratic Programming. 26
MoD Mobility on Demand. 2, 3
MPC Model Predictive Control. 4–9, 15, 30, 32, 37, 43

PACF Partial Autocorrelation Function. 10

SoC State of Charge. xii, 10–12, 15, 21, 25, 27, 28, 40, 42, 43, 48, 50, 52, 61

T-EAMoD Tunable-Electric Autonomous Mobility on Demand. ix, xii, 10, 11, 15, 31, 50–52, 55, 56, 59, 60

WT Waiting time. 12, 52, 53, 55, 56, 60

1

Introduction

Sustainable mobility is an ideal model of a transport system which aims at reducing its environmental impact while maximizing its efficiency and effectiveness. It consists in a series of solutions that contribute to creating a new idea of mobility, enhancing its convenience for both the environment and the passengers. By the dawn of the new millennium, the topic of sustainable transportation has been widely discussed and developed, and it has seen a further advance with the emerge of electricity-powered vehicles and self-driving cars. In the perspective of creating a sustainable mobility system, in fact, it is necessary to both reduce the environmental impact of the transport sector and to make mobility smarter by introducing technology-driven innovations. Electrical vehicles represent nowadays a key tool for providing strong responses to the current environmental, social and sanitary crisis, by leveraging on the adoption of a different form of energy to power the vehicles, leading to a sensible reduction of the harmful emissions connected to vehicles' travels. Autonomous vehicles carry several benefits as well, including: a reduction of the drivers' stress, as they can rest during the journey, time saving, as the riders can spend the ride time doing other things, mobility for non-drivers and an increase in safety. Nowadays, in fact, the majority of car accidents is caused by human errors, the main factors being: drunk driving, speeding, distracted driving and drowsy. The Transport area of the European Commission website [1] claims this percentage to be around 90%, while according to the 2016 research of the NHTSA about fatal motor vehicle crashes, that value would go up to almost 96% [2]. The use of autonomous vehicles could potentially eliminate the majority of the factors that cause car accidents, thus making the roads much safer for both travellers and pedestrians. To achieve all these benefits, vehicles must be able to operate autonomously under all normal conditions, thus reaching the so-called Level 5 autonomy [3]. The benefits of autonomous vehicles are obviously followed by related disadvantages, such as an increase in vehicle and infrastructure cost: for this reason it is still necessary to conduct further researches on the optimization of their functioning. Autonomous vehicles would allow to replace cities' transport systems, including taxis, private cars and public buses, with automated mobility-on-demand systems, thus leading to a reduction of the environmental impact of the transport sector. Such systems would provide affordable mobility and at the same time offer a spatial and temporal flexibility never seen before.

1.1 Mobility on Demand

The advent of private vehicles in the last century has completely revolutionized the nature of urban mobility, as they allow people to easily move either along short or long distances at their discretion. Nevertheless, the exponential growth of this phenomenon in the last decades has led to several environmental issues, such as an excessive release of greenhouse gases and an increasing need of fossil fuels, as well as to several infrastructural issues, namely the lack of parking spots the increase of congestion in the cities. All these drawbacks of private automobiles made them an unsustainable solution to the growing request of personal mobility in the future [4]. Most of the automobiles, nowadays, have the capacity of reaching high speeds and elevated fuel capacity which, combined with a high fuel economy, allows them to travel long distances on a full tank. Studies have demonstrated that in urban environments these features are underutilized, in fact the average cruising speed of vehicles and distance they travel are much lower than their capacity. Moreover, the cars' utilisation rates are typically below 10%, as they spent most of the time being parked [4].

One of the most up-and-coming solutions to these issues is adopting the concept of Mobility on Demand (MoD), consisting of shared vehicles used for one-way passenger mobility. Two categories of services can be included in this transport system modality: taxis and car sharing. The first one involves vehicles with drivers that can reach customers in their location and transport them to their desired destination. This kind of service is provided, besides the common urban taxi suppliers, also by companies such as Uber [5], Lyft [6] and Bolt [7]. Car sharing instead consists in allowing customers to borrow the nearest vehicle, use it to arrive to their destinations and leave it there, thus making it available for the next customers. All these actions are easily done through an application on their smartphone in few seconds. Many companies are providing this kind of service worldwide such as aimo [8], M [9] CAR2GO [10], Share'ngo [11].

1.2 Autonomous Mobility on Demand

A common challenge that must be addressed by the companies providing MoD is that of the imbalance defined as an uneven distribution of the resources among the network. In fact, even though the network is symmetric, being the demand stochastic, it will drive the system out of balance, leading to a periodic accumulation of vehicles in some particular stations and a lack in the others, thus threatening the quality of the service. For companies that provide bike- or scooter-sharing systems, this issue can be solved by moving some of the vehicles from the areas of excess to those of depletion using trucks. For MoD systems relying on the use of cars, a possible solution to the imbalance problem would be to hire a team of drivers, whose job would be to rebalance the vehicles across the network. This option would lead to an increase of expenses in terms of cost and time, and the rebalancers themselves would become unbalanced, thus giving rise to the need of rebalancing also the rebalancers [12].

Driverless vehicles have the potential of solving this issue at its core, as they are able to rebalance autonomously and reach charging spots when needed by themselves. Besides that, a Mobility on Demand system based on the use of autonomous vehicles would provide countless other benefits to urban mobility, which can be classified as external or internal. The internal benefits are directly related to the consumers and their interaction with the transport system, and they can be divided into two subgroups: there are aspects that differentiate AMoD from common taxis, in particular the reduction of the cost of the trip due to the absence of the driver, and others that come from the comparison of AMoD with car sharing services, namely a reduction of physical and mental stress for the driver, considering that he/she can relax without paying attention to the road, and the ability to ride even for customers without driving license.

As for the external benefits, AMoD systems lead to an increase in safety, by eliminating all the human risk factors, a decrease in congestion of the roads, and a solution to the problem of the parking spots, thanks to a more efficient utilization of vehicles. They have also a positive impact on the environment, as they reduce harmful emissions by promoting an increase in fuel efficiency [13].

For what concerns the financial analysis, it has been studied how AMoD affects the fleet size. In particular, *Pavone et al* in [4] show that Manhattan's taxi demand can be satisfied with 30% less vehicles than the current taxi fleet, while the whole personal mobility of Singapore can be fulfilled with only 1/3 of the actual taxis just using AMoD systems.

The last years have seen an acceleration in vehicles automation technologies which, coupled with an increase in public interest in MoD systems, has enhanced the perceived economic and societal value of AMoD systems. This led to a new focus of some automotive giants in the field of robotaxis development, as well as the foundation of new startups with the same scope, such as Zoox [14], owned by Amazon, and EasyMile [15].

As every revolutionary technology, the development of robotaxi systems is fueling heated debates about the perceived risks and drawbacks connected to it, and it is facing a certain amount of skepticism by the public opinion. The main arguments against it involve primarily the economic aspect, as it is perceived to be very expensive for both companies providing it and their potential customers, and safety issues, that are mostly related to the aspect of autonomous driving. Other arguments include: the skepticism about the utility of such new technology, in fact a part of the public has difficulty in seeing why AMoD systems are needed and how they could be better than the current technology; apprehension about the reduction of unskilled employment that such systems would cause, due to the fact that drivers would not be needed anymore; and concerns about passengers' privacy and cybersecurity in general. In conclusion, the numerous challenges that come with the development of AMoD systems include also facing of public's technology adoption resistance, which needs to be addressed in the right way in order to encourage the integration of this technology in the society.

1.3 Related work

Several studies have been conducted in the last decade, regarding the development of Autonomous Mobility on Demand systems as a potential solution for generating an intelligent and efficient transportation system in terms of travel time and fuel savings. In particular, many researches, such as [16] and [17], analyzed the possibility of adopting automated taxi services for urban transport, applying them to simulation scenarios of big cities. Other studies evaluated autonomous on-demand systems as a solution to the problem of transportation in rural areas [18]. In such regions the organization of the mobility faces many challenges, due to the low population density and to the larger and less direct routes, which make a common transportation system inefficient and expensive.

In the past years, several researches have been conducted to explore the possibility of adopting the techniques of Model Predictive Control (MPC) for an intelligent control of an AMoD fleet. The present work takes them as a starting point and a set of guidelines to bring the research forward and explore new aspects and possibilities in the field of autonomous taxi fleets control. In [19] for the first time the optimization of vehicle scheduling in an AMoD system is conducted in a model predictive control framework, and some considerations about vehicle charging are included. The cost function implemented to solve the optimization problem includes two terms: one considers customers' waiting time, while the other takes into account the cost of rebalancing vehicles. The addition of a third term involving charging is also evaluated. The problem is solved as a Mixed Integer Linear Programming (MILP), and the algorithm is shown to experimentally outperform the previous control strategies considered. To obtain these results, a decision variable was assigned to each vehicle. This led to a dependency of the problem size on the dimension of the fleet, thus limiting the real-life applications of the method. A different strategy is adopted in other works, where the complexity of the algorithm is set to be only dependent on the number of stations, and not to the number of vehicles or customers, in order to make it adoptable for larger-scale AMoD systems. This is done for example in [20], where the objective function to be minimized includes the cost that comes from moving vehicles to rebalance them, the cost connected to making the customers wait, and the cost of not servicing customers. This paper also explores the effects of adopting short-term forecasts of customer demand based on past data of the real demand, which are shown to yield significant improvements in terms of customer waiting time reduction. Several research groups encountered the issue of the excessive complexity of the algorithms, which made them either impossible to solve or to scale. Some of the techniques adopted to ease the problem's complexity include decoupling the dispatching and rebalancing actions in order to solve them as two separate unimodular linear programs [21], and relaxing the integer constraint for some of the decision variables in the optimization problem [22]. To the present date, only few works have addressed the aspect of charging of the vehicles. One of these is [23], in which the electricity price and availability variations over time are also taken into account. A two-layers MPC optimization problem is implemented, with different time frames: the charging optimization problem has a longer horizon, while the optimization of the transport service happens within a shorter time frame, and takes as constraints the results of the other problem. Two more elements are added in the cost function: the state of charge of the vehicles and the charging rate. The results obtained in the study are very positive in terms of reduction of the charging cost, but the problem is limited in scale.

The aspect of charging in the research on AMoD control needs to be further investigated and properly simulated, in fact in a complete model it is essential to account for it. Furthermore, solutions that implement a predictive framework, such as the MPC optimization problem, need a reliable forecast of the demand on which additional studies are required.

1.4 Objectives of the study

In order to see the real benefits of the adoption of a fleet of autonomous vehicles for the purpose of passenger transportation, it is crucial to operate an appropriate control of the system that grants it an efficient and fully optimized performance. The control of an autonomous taxi fleet can be divided into routing, dispatching and rebalancing, and here only the latter two parts will be addressed. The dispatching part consists of matching the open customer requests with available vehicles. The problem of imbalance is a drawback of the system's high flexibility, and could lead to drastically decreased service levels: it is thus necessary to periodically reposition vehicles from oversupplied to empty areas of the city [16].

The main questions to which this research aims at providing answers are:

- Which factors should play a major role in the assignment of vehicles to customers in an AMoD system, and what is their impact on the optimality of the solution?
- How does the rebalancing of resources along the network behave in a predictive framework?
- How can the energy consumption of the AMoD system be optimized?
- How well can the control policy scale to the complexity and dimensions of real systems?

The aim of this research is then to create an optimal algorithm to control the behavior of a taxi fleet for AMoD, with the objective of maximizing its efficiency in terms of energy consumption while keeping customer waiting time under control. In order to achieve a more efficient algorithm, a Model Predictive Control (MPC) approach has been used in the rebalancing phase, which is thus approached as an optimization problem with a receding horizon. Moreover, as suggested in [24], this study introduces information about the desired destinations of the present and future customers in order to improve the quality of the dispatching and rebalancing decisions: previously, the actions issued by the controller were planned only basing on the customers' sources, preventing to obtain a fully optimized solution. The aspect of vehicles charging is also introduced, to be addressed here in a pro-active way in order to not penalize the system and to adjust to its necessities.

1.5 Thesis Organization

After this introduction on the subject, in Chapter 2 there is an explanation of all the theory behind this research. The topics addressed include: the Hungarian algorithm adopted to solve the dispatching problem, the techniques of Model Predictive Control used in the rebalancing phase and the Autoregressive forecasting method adopted to predict the future customer demand. Lately, in the same chapter, the algorithm created in this study is explained in detail, comprehensive of the three phases that compose it. Chapter 3 follows, which exposes how the algorithm was implemented and tested, and the results that were extracted from it. This chapter involves two different kinds of simulations. The first is performed on a fictitious small network: it is useful mainly to extract theoretical results about the meaning of different parameters and investigate their optimal values, as well as to test the behavior of the whole algorithm. Here, the three phases that compose it are separately addressed and clarified with the aid of some examples. Afterwards, a network of the City of Chicago was used to simulate the algorithm, providing realistic results in terms of distance travelled by taxis and time that the customers need to wait for the service. Finally, Chapter 4 contains a broader discussion on the results previously exposed, as well as a presentation of the possible extensions of the study that would be interesting to investigate in the future. In this way, the conclusion to the whole research are finally drawn.

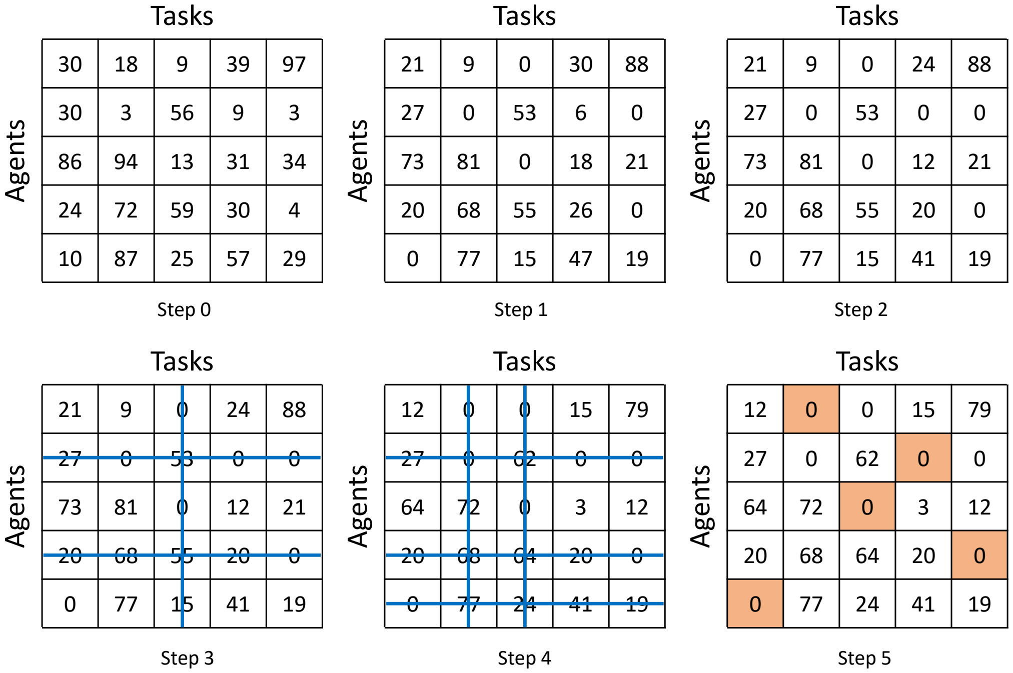
2
Theory

In this chapter, the theoretical methods that constitute the basis of this thesis are addressed. These include an explanation of the Hungarian algorithm, of the Model Predictive Control techiques, and of the Autoregressive forecasting method, which were all adopted in the development of the new algorithm that represents the outcome of this research. Finally, such algorithm is presented and explained in detail in the last section of this chapter.

2.1 Hungarian method

The Hungarian algorithm, also knows as Kuhn-Munkres algorithm, is a combinatorial optimization method used to solve the assignment problem, an operational research problem which consists in having to assign different resources to tasks in an optimal way, thus minimizing the total cost. The assignment problem is expressed through a bipartite graph, represented by an adjacency matrix $X \in \mathbb{R}^{n \times m}$ where the element in the i-th row and j-th column represents the cost of assigning the i-th task to the j-th resource. If the problem is balanced, meaning that the number of tasks is equal to the number of resources ($m = n$), the opportunity cost matrix is square, otherwise some dummy rows or column can be added to make it square. The procedure that leads to solving it is based on a theoretical principle stating that if an optimal assignment to the cost matrix exists, then it exists also for a matrix obtained by subtracting or adding a number from all the entries of a row or a column of the original matrix, and the optimal assignment in the two cases is the same. The steps of the Hungarian algorithm, showed through an example in Figure 2.1, consist in:

1. finding the smallest element for each row and subtracting it from all the elements of the same row,
2. finding the smallest element for each column and subtracting it from all the elements of the same column,
3. covering with a line all the rows and columns that contain at least one zero element: if the number of lines is equal to the size of the matrix, then a result has been obtained, otherwise the next step takes place
4. finding the smallest element not covered by a line: subtracting it from all uncovered elements, and adding it to all the elements covered by two lines,
5. repeating the last two steps until the number of covering lines equals the size of the matrix. The assignment is then performed by pairing rows and columns whose common element is a zero.



30	18	9	39	97
30	3	56	9	3
86	94	13	31	34
24	72	59	30	4
10	87	25	57	29

Step 0

21	9	0	30	88
27	0	53	6	0
73	81	0	18	21
20	68	55	26	0
0	77	15	47	19

Step 1

21	9	0	24	88
27	0	53	0	0
73	81	0	12	21
20	68	55	20	0
0	77	15	41	19

Step 2

21	9	~~0~~	24	88
~~27~~	~~0~~	~~53~~	~~0~~	~~0~~
73	81	~~0~~	12	21
~~20~~	~~68~~	~~55~~	~~20~~	~~0~~
0	77	~~15~~	41	19

Step 3

12	~~0~~	~~0~~	15	79
~~27~~	~~0~~	~~62~~	~~0~~	~~0~~
64	~~72~~	~~0~~	3	12
~~20~~	~~68~~	~~64~~	~~20~~	~~0~~
~~0~~	~~77~~	~~24~~	~~41~~	~~19~~

Step 4

12	0	0	15	79
27	0	62	0	0
64	72	0	3	12
20	68	64	20	0
0	77	24	41	19

Step 5

Figure 2.1: *Example of the Hungarian algorithm.*

The Hungarian algorithm has a polynomial run-time complexity of $O(n^3)$ in the worst case, meaning that the computational burden in large scale matrices can become prohibitive [25]. For this reason, it was necessary to implement several actions to reduce the computational complexity of the algorithm and make it scalable and thus adoptable to real systems.

2.2 Model Predictive Control

The term Model Predictive Control (MPC) refers to a category of control strategies based on the use of internal models of the plants, in which an open-loop optimization problem is solved at each time step to produce a sequence of control actions calculated up to a fixed time horizon. A schematic representation of the MPC loop is shown in Figure 2.3. To leverage control inputs, the optimization uses predictions based on the plant model for the whole horizon: the presence of the horizon provides a tradeoff between short-term and long-term benefits. Subsequently, only the first control action is applied as the input signal to the plant: the idea is to select the input which promises the best predicted behavior. Then, one sampling interval later the whole cycle of output measurements, predictions and input trajectory determination is repeated. As shown in figure 2.2, the length of the horizon remains the same, but it slides along by one sampling interval at each time step: for this reason, this control technique is also addressed as "receding horizon control". The main advantages of this strategy are the possibility to easily deal with nonlinear systems with complex objectives and multiple variables, ensuring profit maximization and

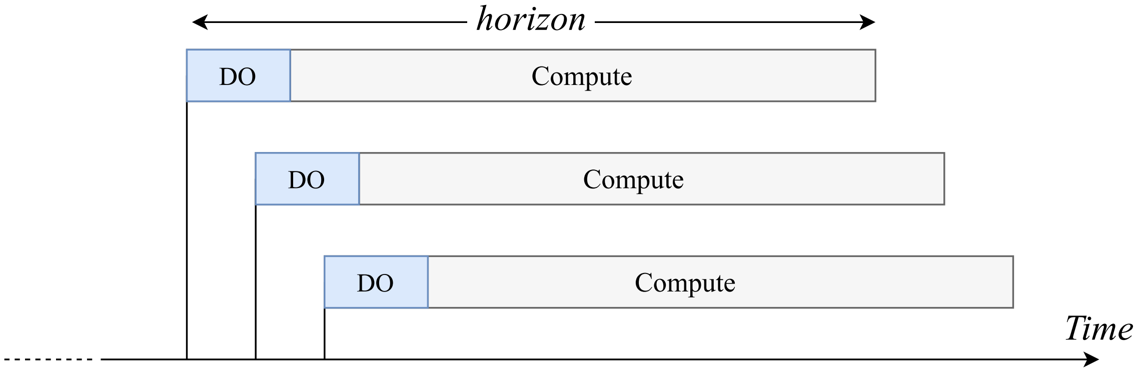


Figure 2.2: *Visual explanation of the receding horizon technique.*

performance optimization, and its capability to explicitly handle the constraints of the system. In fact, in classic control techniques, it is usually preferable to not operate a plant exactly at the real limits of its capabilities, because in that way unexpected disturbances from various sources could lead the system out of its constraints, but at the same time adopting an extremely conservative approach would lead to a reduction of efficiency. MPC solves this problem, in fact the controller is aware of the input constraints because they are included in the model design, and thus it can better handle the different kinds of constraints, leading to the possibility of operating the plant at its optimal set-point, even when it is very close to the constraint [26]. The drawbacks of the MPC techniques include mainly their high computational cost and coding complexity.

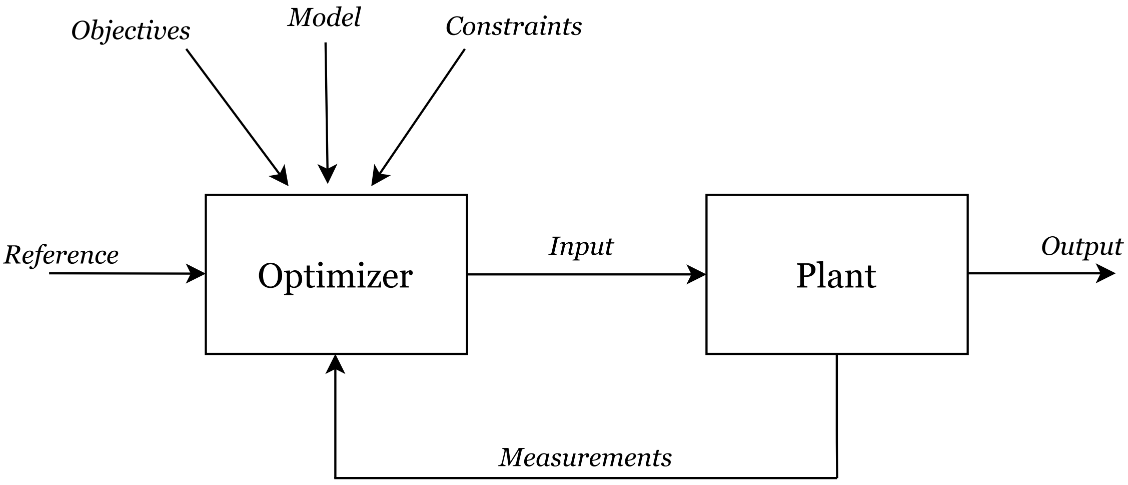


Figure 2.3: *Block scheme of MPC.*

2.3 AR models

A regression model predicts the values of an output variable of a stochastic process basing on a linear combination of input variables, using coefficients obtained by optimizing the model on training data. An Autoregressive (AR) model is a particular case of a regressive model where past values of the variable of interest are used as inputs to the system, along with a stochastic term indicating the number of regressors to be used in the simulation. An AR is in fact a tool for generating time series forecasting of future data basing on past values, relying on the assumption that older data have an effect on future values. Such relation between input and output variables is called correlation, and in particular autocorrelation in the case of autoregressive models: it is positive if an increase of one kind of variables corresponds to an increase in the other, and negative if the variables have opposite behaviors. A further extension of the concept of correlation is the Partial Autocorrelation Function (PACF), which instead allows to evaluate the relation between a current observation and a past observation corresponding to a specific lagged value, without accounting for values at shorter lags: the effect of any correlation caused by terms at shorter lags is removed. By studying the behavior of these statistics, it is possible to evaluate the dependence of the variables on each other and consequently select the order of the model, which corresponds to the optimal number of immediately preceding values in the series that are used to compute the forecasts (lagged values) [27]. The appropriate maximum number of lags is in fact selected as the order of the last lag for which the value of the PACF is different from zero: beyond it all the partial autocorrelations are null. An autoregressive model of order p can thus be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{2.1}$$

where $\phi_1, ..., \phi_t$ are the parameters of the model, estimated through the least-squares optimization or other more complex procedures, c is a constant and ϵ_t is a white noise term [28]. A one-time shock in an AR model, consisting of a nonzero value of ϵ_t at a certain time step, will affect the predictions of the values of the variable y infinitely far into the future [29].

2.4 Developed Algorithm: T-EAMoD

The algorithm created in this research consists of three main steps, mainly: the static assignment, the dispatching and the rebalancing. At the beginning of each time step, an evaluation of the demand is first executed, in which new customers are summed to the old ones who were not serviced before. In this way, a precise notion on the total amount of customers presents in the network is available, as well as their exact positions, desired destinations and waiting times. The network is composed of stations, seen as the specific locations where taxis and customers can be located, connected by roads, where vehicles can travel, either transporting passengers or empty. In the scenario for which this algorithm is designed, the taxis are electric and charging can happen in every station: the charging capacity of the stations is not taken into account. The necessity of charging a vehicle in general rises when its battery reaches a fixed lower threshold: in this way its State of Charge (SoC) should never go to zero. The unplugging from the charging station instead happens when the vehicle's SoC gets to a certain higher threshold. The algorithm developed here is addressed as "Tunable-Electric Autonomous Mobility on Demand

(T-EAMoD)", where the term "Tunable" is used to highlight the fact that the many parameters which play a role in the algorithm can be easily adjusted in order to adapt the policy to different possible objectives, according to the need.

2.4.1 Static Assignment

Having obtained precise information about all the customers that want to benefit from the service, the static assignment step takes place. It consists in the best-case assignment, and it happens when a taxi and a customer are in the same station. If the battery level of the taxi is enough to complete the trip that the customer is requesting, then the two are matched, and they soon disappear from the station. The taxi will then re-appear as available in the passenger's destination station a number of time steps later equal to the time needed to complete such trip. The waiting time of a customer assigned in this way is almost zero (or no more than a sampling period), and the taxi does not need to travel empty because the customer is in its exact same position. For these reasons, this kind of assignment results very beneficial to the overall performance of the system, so it is necessary to put it into practice as much as possible. To do so, the option of reactively unplugging the taxis that are currently charging is available. This means that, after assigning all the free taxis, if there are still unserviced customers in a station and in the same location there are taxis that are charging, some of them can be unplugged and assigned to the residual customers. This can happen only after checking the battery levels of the charging taxis and making sure it is higher than what is needed to transport the customers to their desired destination.

2.4.2 Dispatching

The dispatching phase comes right after matching all the customers to the taxis available and with enough battery that are located in their same stations. It consists in running the Hungarian algorithm for all the residual customers and all the taxis in the network. All the variables used to evaluate the matrix are reported in Table 2.1, while the method used to calculate each value is described in Algorithm 1.
A $N_T \times N_C$ cost matrix is created, where N_T is the total number of taxis in the network, while N_C indicates how many customers still need to be assigned. The c_{ij} resource-task matching cost is constituted of multiple components, the main one being the cost, for the taxi, of travelling to cover the distance that separates it, or its last destination (if already assigned), from the customer. At the same time, the battery of the taxi (SoC_i) is compared to the battery needed to cover the trip requested by the customer plus the distance to pickup the customer (SoC^*_{trip}). If the taxi, according to its SoC, is not able to fulfill the requested trip, then the c_{ij} element of the matrix is set to an infinite value, to ensure that assignment will never take place. Otherwise, an additional cost factor inversely proportional to its SoC is added. In this way, if the state of charge is very low, its assignment cost will increase. Different other cost factors come into play, according to the state of the taxi. For each taxi that is already on duty, an additional cost contribution (P_{res_i}) must be added, to account for the queue of customers that it has to serve before heading to

N_S	Number of network's stations
N_T	Number of fleet's taxis
N_C	Number of unassigned customers
c_{ij}	Matching cost for the assignment of the i-th taxi to the j-th customer
i_{pos}	Position of the i-th taxi
i_{last_dest}	Last destination of the i-th taxi, valid only if it is assigned already
j_{pos}	Position of the j-th customer
P_{ij}	Length of the path between i-th taxi and the j-th customer
P_{res_i}	Remaining path of the i-th taxi, valid only if it is assigned already
WT_j	Waiting time of the j-th customer
SoC_i	State of Charge of the i-th taxi
SoC^*_{trip}	State of Charge needed to fulfill the whole trip (P_{ij} + j-th costumer trip)
c_s	Starting cost
MAX_P	The longest path among all the taxi-customer combinations present at the moment
MAX_{WT}	The longest customer's waiting time
MAX_{SoC}	Maximum level of battery of the fleet
$w_P = 0.65$	Weight of the taxi's path length
$w_{WT} = 0.15$	Weight of the customer's waiting time
$w_{SoC} = 0.1$	Weight of the taxi's State of Charge
$w_s = 0.1$	Weight of the taxi's starting cost

Table 2.1: *Components of the cost matrix element.*

Algorithm 1: Cost matrix construction for dispatching

Input: taxis (N_T), customers (N_C), cost components, weights

Output: Cost matrix $\in \mathbb{R}^{N_T \times N_C}$

```
for $i$ in $N$ do
    for $j$ in $M$ do
        if $SoC_i > SoC^*_{trip}$ then
            if $state_i$ is stay or $state_i$ is charging then
                $P_{ij}$ = shortest_path(from $i_{pos}$ to $j_{pos}$)
                $c_{ij} = \frac{P_{ij}}{MAX_P} \cdot w_P + \frac{MAX_{WT} - WT_j}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC} - SoC_i}{MAX_{SoC}} \cdot w_{SoC} + c_s \cdot w_s$
            else
                $P_{ij}$ = shortest_path(from $i_{last\_dest}$ to $j_{pos}$)
                $c_{ij} = \frac{P_{ij} + P_{res_i}}{MAX_P} \cdot w_P + \frac{MAX_{WT} - WT_j}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC} - SoC_i}{MAX_{SoC}} \cdot w_{SoC}$
            end
        else
            $c_{ij} = \infty$
        end
    end
end
```

the new customer. For the taxis that are not moving, a small contribution (c_s) is summed as well, to account for the cost of starting the vehicle when it is off. This cost element is added in order to prioritize, in case of equal total distance of two vehicles from a customer C_1, the taxi that is already moving over the one that is not. Such situation can only occurr when the customer that is already travelling is transporting a person whose destination corresponds to the station where the unassigned customer C_1 is located, or is at least very close to it. It would then not be necessary to start the vehicle that is not moving and send it to the same station where the moving one is already going, in case the time it would take to reach such station is the same for both. The inclusion of the starting cost would then allow to avoid the energy expense that would come from moving an idle vehicle where it is not needed.

In the dispatching phase, the charging is reactive, meaning that even the taxis that are in charging state are considered suitable for the assignment, and thus included in the cost matrix. If the dispatcher selects a charging taxi as the optimal resource to be assigned to a customer, the vehicle is then reactively unplugged and sent to pick up the person.

To prioritize customers according their waiting time, a supplementary factor is added to the resource-task matching cost c_{ij} corresponding to them. In this way, the customers that have been waiting for longer will be served first.

All the costs mentioned are normalized over the respective maximum value that each of them can assume, and then they are multiplied for appropriate weight factors that define how much impact each of them should have over the dispatching decision. This action ensures that the cost corresponding to each possible resource-task matching is comparable to those of the other options, and that the solution chosen is the optimal one. The weights assigned to the different cost components are summarized in table 2.1. In future applications of this algorithm, they can be appropriately tuned according to the desired objective: if, for example, the main aim of a service provider is to satisfy first the customers that have been waiting for longer, without worrying much about the energy expense, it is possible to increase the weight w_{WT} while decreasing w_P of the same amount. In the context of this study, the choice of the values of the weights reflected the intention to reduce the overall energy consumption: for this reason, the weight that accounts for length of the trips that the taxis have to do to reach each customer, which is directly connected to the energy expense of the vehicles, was assigned a higher value. The second objective involves the quality of the service in terms of customer waiting time, and for this reason the weight connected to such aspect is higher than the remaining ones, which instead only make a difference in the cases where the lengths of the paths of different taxis and the customers' waiting times are already comparable to each other.

At the end of the dispatching phase, the maximum number of assignments that has been made corresponds to the minimum value between the amount of customers and that of the taxis, thus in the best case scenario either all the customers or all the taxis in the network are assigned.

2.4.3 Rebalancing

All the taxis that still remain available after the two previous stages undergo the third and last phase of the T-EAMoD algorithm: the rebalancing. It consists in providing empty vehicles with control actions aiming at redistributing them across the whole network, in order to have taxis available in the stations where they will be most needed in the future. This is done by controlling the vehicles in a predictive framework, thus applying the techniques of discrete MPC. A control sequence is obtained at each time step by solving an optimization problem defined by: the objective to be minimized, the model which predicts the system's behavior, and the constraints that need to be satisfied. The first control action in the sequence is then applied to the plant. The steps of the whole rebalancing process are summarized in Algorithm 2.

Algorithm 2: Rebalancing

```
Input: network ($N_S$), horizon ($H$), taxis ($N_T$), batch
Output: $X^0$ , $Z^0 \in \mathbb{N}^{N_S \times N_S}$
for each rebalancing time step do
    $\Sigma$ = taxi_path_evaluation(network, taxis)
    $\Phi$ = forecaster(batch, $H$)
    Solve optimization problem to obtain $X$, $Y$, $Z$
    for $i$ in $N$ do
        for $j$ in $N$ do
            if $x^0_{ij} > 0$ then
                Rebalance $x_{ij}$ taxis with the highest SoC from station $i$ to
                  station $j$
            end
        end
        if $z^0_i > 0$ then
            Put in charge $z_i$ taxis with the lowest SoC in station $i$
        end
    end
end
```

Demand forecast

Since the rebalancing algorithm relies on the technique of Model Predictive Control, it looks ahead in the future for a number of time steps equal to the size of the horizon H. In order to successfully adopt this method, it is necessary to foresee the presence of customers in each station of the network, for H time steps ahead of the present one: this is done by implementing an Autoregressive (AR) model. For each instant, a batch of data from the previous time steps is used as input to create the AR model for each station. In absence of real data, sequences of fictitious demand are generated in order to simulate customers arrival for a full day and a full week. The demand data include information about the position of each customer, as well as its desired destination, and the forecaster will produce results that involve both as well. To roughly evaluate the optimal number of regressors for each forecasting action, the model needs first to be trained on previous data, for which the partial autocorrelation between different values is studied. Eventually, a further analysis is done to validate or improve the choice of regressors, basing on the performance of the forecaster compared to the train data. For each couple of stations and each time step of the day, the deviation between the average real demand and all the average forecasts obtained using different numbers of regressors is computed, and the number of regressors leading to the smallest error is chosen and saved, in order to use it in the simulations.

To evaluate the effectiveness of the AR forecaster, as well as the sensitivity of the system model to the quality of the forecasted information, it is possible to compare the case where the AR is used with others where the model can be either subjected to a predefined fixed demand, that also substitutes the forecasts for the future, or to the same demand of a certain time step for the whole horizon.

Cost function

The objective function to be minimized is constituted of three components, which represent opportunity cost sources connected to the rebalancing actions, mainly: the energy expense of the rebalancing trip, the penalty given by the presence of an imbalance between the amount of taxis and customers in each station, the opportunity cost coming from the decision of charging some vehicles.

The three decision variables of the rebalancing problem, directly connected to the cost components, are:

- x_{ij}^t, which indicates the number of taxis sent by the rebalancer from station i to station j at time t. X is a 3D-matrix whose dimensions depend on the length of the time horizon (H) and the square of the number of stations (N_S). The value of x_{ij}^t can be nonzero only if the two stations i and j are adjacent: if a direct edge between them does not exist, in fact, the rebalancing cannot happen;
- y_i^t, the imbalance between the number of taxis and customers in station i at time t. Y is a 2D-matrix, with size given by the length of the horizon (H) and the number of stations (N_S). Being defined as the difference between the number of customers and that of the taxis, its elements can be either positive or negative. A positive imbalance indicates an excess of customers in the station, that can't be satisfied at the moment being, while a negative value of y means that there are more taxis than needed in the station, so some of them could be easily sent away or charged;
- z_i^t, representing the number of taxis to put into charging state in station i at

time t. Z is a 2D-matrix whose dimensions depend on the length of the horizon (H) and on the number of stations (N_S), and its values can only be positive. In this way, the rebalancer has the option to put vehicles into charging state when there is an excess of them in a station, taking advantage of those hours when they are less needed to restore their battery, in order to both reduce the imbalance and also have taxis ready for the peak hours. Not all the vehicles expressed in z are necessarily put in charge, in fact a selection will be done basing on their battery level: z represents just an upper bound to the number of taxis that can be plugged.

The expression of the cost function to be minimized is then:

$$J = \sum_{t=0}^{H}\sum_{i=1}^{N_S}\sum_{j=1}^{N_S} w_{x_{ij}} x_{ij}^t + \sum_{t=1}^{H}\sum_{i=1}^{N_S} w_{y_i}(y_i^t + \widetilde{y_i}^t)^2 + \sum_{t=0}^{H}\sum_{i=1}^{N_S} w_{z_i}^t z_i^t \tag{2.2}$$

Here, N indicates the total number of stations in the network, while H corresponds to the length of the time horizon adopted.

The value of $\widetilde{y_i}^t$ indicates the steady state equilibrium point of the system. This term is equal to the amount of taxis that should be present in each station when the system is in equilibrium. It is useful mostly in the cases where customers are absent or there are very few of them: an equilibrium condition can be re-established by spreading vehicles along the network, with a fixed different amount of them in every station indicated by the values of $\widetilde{y_i}^t$. In this study, the value of $\widetilde{y_i}^t$ was always set to zero, because the focus is on managing in the best possible way the cases with high demand. For a proper use of this parameter, more accurate studies about the optimum steady state condition of the system would be needed. To simplify the notation, from now on the term $(y_i^t + \widetilde{y_i}^t)^2$ will be indicated as $(y_i^t)^2$ or just y^2.

The value of the imbalance in the expression is squared in order to eliminate the negative sign. The difference between number of customers and taxis in a station, in fact, needs to be as close as possible to zero. If the imbalance term in the cost function is not squared, the optimizer will just tend to increase the number of taxis where they are not needed: y would assume a negative large-module value. This would easily lead to a decrease in the value of the cost function, but it would correspond to a meaningless result. To solve the just mentioned issue and to reduce the computational cost that the square brings, another version of the cost function using the absolute value of y has been studied. The results obtained with this cost function will be analyzed in Section 3.2.

Furthermore, as can be seen from Equation 2.2, the y-terms associated to time step 0 of the horizon are not included in the cost function of the optimization problem. The results of this choice will be discussed more in detail in Section 3.2, although the idea behind it is that the current situation should not affect the decisions taken.

The weights in the cost function are chosen in order to define the incidence that each cost component should have on the decision-making process of the rebalancing phase. The weight matrices W_x, W_y and W_z have the same dimensions of respectively X, Y and Z. The selection of W_x and W_y, respectively associated to the cost of rebalancing vehicles and the cost of the imbalance, was object of an accurate study and will be discussed more in detail later in this section. The weight connected to the charging of the vehicles W_z, instead, is always set to zero. The cost components related to the action of charging the taxis, in fact, are not differential: charging is not an option, it needs to be done in any case, either sooner or later. And since this study does not take into account the fluctuations of the electricity price and availability, there will not be any cost difference in plugging a vehicle to a charging station in different times of the day. The term z was still included in order to make the cost function more general and complete: in this form it could also be adopted in future studies aimed at investigating the charging aspect in a more detailed way. In that case it would just be necessary to evaluate the right values for the weights W_z, which would not be zero anymore.

Constraints

The constraints adopted in the optimization problem ensure that the decision variables x and z always assume integer positive values, in fact they represent the amounts of taxi that have to do certain actions, so negative or non-integer values would be meaningless in their case. For what concerns the decision variable y, it was defined as an algebraic sum of all integer components, but it was not constrained to be integer itself in order to avoid an unnecessary increase in the optimizer's effort: it was in fact defined as a continuous variable.
The other constraints are created to give an upper bound to the amount of outgoing and charging taxis in each station at every time step, and to properly define the imbalance as the difference between the amount of customers and taxis in each station in every moment, considering all the existing contributes. The variables used to define the constraints are summarized in Table 2.2.

For what concerns the imbalance, in each time step it is computed as the difference between the net number of customers in each station and the net number of taxis available in the same place.
The time step 0 of the horizon represents the current time step, thus all the variables referring to it are either variables of the present state or control inputs that will be sent to the network in the end of the evaluation. Consequently, the imbalance of time step 0 of the horizon for station i, showed in Equation 2.3, includes only the contributions given by: the remaining demand at time 0 after the static assignment and dispatching steps ($\phi^0_{i_{source}}$), the available taxis (σ^0_i), the taxis to put in charge (z^0_i) and the outgoing taxis ($x^0_{i_{out}}$).

$$y^0_i = \phi^0_{i_{source}} - \sigma^0_i + z^0_i + x^0_{i_{out}} \qquad \text{for } i = 1, ..., N_S \tag{2.3}$$

In a scenario with zero demand within the whole network, a station where taxis are present has a negative imbalance. Since the weight related to the charging cost is always zero while rebalancing a taxi has a non-zero cost, the optimizer will try to increase z_i instead of $x_{i_{out}}$ to minimize the overall cost. This action leads to a more efficient management of the fleet avoiding unnecessary energy utilization.
Furthermore, the imbalance of the stations in the remaining time steps of the horizon has to include all the previous decisions. This constraint, as can be seen in Equation 2.4, is defined in a recursive way within the horizon, to establish a direct

N_S	Number of stations
H	Length of the horizon in time steps
y_i^t	Imbalance of the station i at the time step t
$\widetilde{y}_i^t$	Desired steady state equilibrium
σ_i^0	Number of taxis currently available in the station i
σ_i^t	Number of taxis becoming available in the station i at the time step t because either their passenger's trip ends there or they have completed their charging phase
$\phi_{i_{source}}^0$	Number of customer with station i as source not assigned yet
$\phi_{i_{source}}^t$	Number of forcasted customer with station i as source at the time step t
$\phi_{i_{dest}}^t$	Number of taxis that, according to the predictions, took a customer at time $t - t_{path_{ji}}$ whose destination was i, where $t_{path_{ji}}$ is the time needed to travel the distance from station j to station i. They arrive at time step t in the station i
x_{ij}^t	Number of taxis rebalanced from station i to station j at time step t
$x_{i_{in}}^t$	Number of rebalancing taxis entering in station i at time step t (Equation 2.6)
$x_{i_{out}}^t$	Number of rebalancing taxis leaving the station i at time step t (Equation 2.5)
z_i^t	Number of taxis to put in charge in the station i at the time step t
$z_{i_{end}}^t$	Number of taxis becoming available in station i at time step t because they were put in charge at time step $t - delay$, where $delay$ is the average charging duration of a vehicle (Equation 2.7)
k_i^t	Number of taxis in the station i at the time step t whose battery is below a certain threshold

Table 2.2: *Variables used in the constraints of the optimization problem.*

correlation between the different time steps: in this way, the value assigned to the decision variables in a precise moment has clear effects on the next steps of the horizon. This ensures that the solution found is optimal not for the present moment but in the long-term.

$$y_i^t = y_i^{t-1} - \sigma_i^t + x_{i_{out}}^t - x_{i_{in}}^t - z_{i_{end}}^t + z_i^t + \phi_{i_{source}}^t - \phi_{i_{dest}}^{t-path} \tag{2.4}$$

$$\text{for } t = 1, ..., H \text{ and } i = 1, ..., N_S$$

The imbalance at a certain time step for a specific station y_i^t is the algebraic sum of the imbalance at the previous time step, y_i^{t-1}, and different other terms, whose meaning needs to be clarified.

The term σ_i^t indicates the number of taxis that become available at time t because either their trip or their charging phase ended. This is a well defined and consistent number of taxis evaluated outside the horizon, because it depends on the taxis that are on duty or in charging state, whose duty's ending times are known.

The expression $x_{i_{out}}^t$ represents the rebalancing taxis that are sent out from station i at time step t, showed in the equation 2.5:

$$x_{i_{out}}^t = \sum_{j=1}^{N_S} x_{ij}^t \tag{2.5}$$

while $x_{i_{in}}^t$ indicates the rebalancing taxis that in previous time steps were sent towards station i, and consequently arrive there at time step t, described in Equation 2.6:

$$x_{i_{in}}^t = \sum_{j=1}^{N_S} \sum_{\tau=0}^{t-path_{ji}} x_{ji}^\tau \tag{2.6}$$

where $path_{ji}$ is a matrix that describes the average time required for a taxi to travel from station j to station i.
The reason why both x_{in} and x_{out} appear in the constraint equation also reflects the fact that rebalancing can only happen between neighboring stations. It is in fact a flow movement of taxis through stations, until they reach the forecasted customers: the imbalance of those stations where the taxis are only transiting does not need to contribute to the overall imbalance cost. If a taxi transits through station i at time t, in the computation of the imbalance y_i^t it will count as +1 in the x_{in} and +1 in the x_{out}, thus not affecting the net imbalance in the station at all. Example 2 in Section 3 better clarifies this behavior.

The term z_i^t consists in the number of taxis that have to be put in charge in station i at time step t, while $z_{i_{end}}^t$ represents the number of taxis that were previously put in charge and that, at time step t, are available again, described in Equation 2.7:

$$z_{i_{end}}^t = \sum_{\tau=0}^{t-delay} z_i^\tau \tag{2.7}$$

where *delay* is the average time that a taxi spends in charging, considering that vehicles put in charge by the rebalancer usually already have a residual SoC.

The expression $\phi^t_{i_{source}}$ indicates the number of predicted customers whose source corresponds to station i, while $\phi^t_{i_{dest}}$ represents the number of taxis that took a customer in the previous time steps and are expected to arrive at time t in station i. Here, for the latter term, a simplifying assumption is adopted, by stating that customers are picked up at the same moment they show up in a station. Including $\phi^t_{i_{dest}}$ in the constraint equation is crucial in order to see the positive effects of increasing the horizon length. Without this variable, in fact, when the length of the horizon exceeds the length of the customers' trips, the system becomes unstable and an increase in waiting time and empty distance can be observed. When the demand forecast predicts the arrival of customers in a station, in fact, those are subsequently counted as positive imbalance terms, while the available taxis that supposedly pick them up in the same station, with a predicted static assignment, are subtracted from the count of the imbalance (see Equation 2.4). Neglecting the term $\phi^t_{i_{dest}}$, these taxis would disappear from the computation when assigned, and they would never re-enter in any of the successive time steps: it would then not be possible to re-assign the same taxi more than one time along the whole horizon. The introduction of the forecast of the destinations solves this problem. For instance, if the forecaster predicts a customer who appears at time step t whose source and destination are respectively i and j, this counts as positive imbalance in the variable $\phi^t_{i_{source}}$, where it indicates a request that has to be served, and as negative imbalance in the variable $\phi^{t+t_{path_{ij}}}_{i_{source}}$. In this equation, the term $t_{path_{ij}}$ indicates the average number of time steps needed for a taxi to drive from station i to station j. In this way, when it has completed its trip, the taxi appears in the station corresponding to the destination of its passenger, and becomes available to pickup another customer.

The sum of the amount of taxis that can leave station i at time step t of the horizon ($x^t_{i_{out}}$, see Equation 2.5), and the number of vehicles to put in charge at the same time and station (z^t_i) is limited by the algebraic sum of various contributions, as defined in Equations 2.8 and 2.9. At the first time step of the horizon, indeed the only one for which the control actions computed will be actually executed, this upper bound is given by the amount of taxis that are currently available there, σ^0_i, as shown in Equation 2.8:

$$x^0_{i_{out}} + z^0_i \leq \sigma^0_i \qquad \text{for } i = 1, ..., N_S \tag{2.8}$$

For the residual time steps of the horizon, the constraints need to take into account also all the decisions taken in the previous time steps. To do so, the upper bound is defined for each station as the sum of the taxis that ever became available until that time step (σ_i, $x_{i_{in}}$, $z_{i_{end}}$, $\phi_{i_{dest}}$) decreased of a quantity equal to the amount of taxis sent out the station during all the previous time steps ($x_{i_{out}}$, z_i). The constraint is described in Equation 2.9

$$x^t_{i_{out}} + z^t_i \leq \sum_{\tau=0}^{t-1}(\sigma^\tau_i + x^\tau_{i_{in}} + z^\tau_{i_{end}} + \phi^\tau_{i_{dest}} - x^\tau_{i_{out}} - z^\tau_i) + \sigma^t_i + x^t_{i_{in}} + z^t_{i_{end}} + \phi^t_{i_{dest}} \tag{2.9}$$

$$\text{for } t = 1, ..., H \text{ and } i = 1, ..., N_S$$

A last constraint equation is adopted to regulate the amount of taxis that are set to charging state at each time step. As shown in Equation 2.10 in fact, the maximum number of taxis to be plugged must be lower than a certain threshold k_i^t defined by the amount of vehicles in the station whose battery is below a given value.

$$z_i^t \leq k_i^t \tag{2.10}$$

Weight analysis

For the selection of the weight matrices W_x and W_y, a multi-objective optimization problem was implemented: the service factor γ was introduced, useful to tune the impact that each cost term has on the rebalancing decision. Each weight element was in fact defined as the product of γ with a component that normalizes its cost function term. In this way the range of value that the two costs x and y^2 can assume is restricted to vary between 0 and 1, so it is possible to define their relative importance in relation to each other by varying the service factor γ.

Each $w_{x_{ij}}$, element of the matrix containing the weight of the rebalancing trip from station i to station j, is given by:

$$w_{x_{ij}} = \frac{d_{ij}}{d_{max} \cdot n_{taxi}} \cdot (1 - \gamma)$$

where d_{ij} is the distance between the stations i and j, while d_{max} is the largest possible distance between two stations in the network. Multiplying d_{max} with the total number of taxis then it is possible to obtain the maximum value that the x could ever assume, corresponding to the worst case scenario where all taxis move along the longest edge in the network at the same time.

For what concerns the imbalance weight, each element of the vector is defined as:

$$w_{y_i} = \frac{1}{[max(n_{taxi}, n_{new\_customers})]^2} \cdot \gamma$$

because the maximum magnitude value that the imbalance can assume is one of the following:

- $n_{new\_customers}$, the highest amount of customers that ever appear in the same time step, reflecting the worst case scenario of all customers appearing together in the same station;
- n_{taxi}, the total number of taxis in the network, reflecting the case where all taxis are in the same station without any customer.

Since $n_{new\_customer}$ is typically not known in advance, n_{taxi} was used as maximum magnitude of the imbalance. This assumption is reasonable because, choosing a rebalancing period short enough, in most of the cases it is actually true that $n_{taxi} > n_{new\_customer}$, because customers will not have enough time to accumulate in a station.

With this method it was possible simulate various scenarios, in which different importance was given to the two cost components in the evaluation of the rebalancing decision, and the different cases were compared.

Optimization problem

Eliminating the zero terms and expressing the weights in an explicit way, the cost function can then be rewritten in the following way:

$$J(x,y) = \sum_{t=0}^{H}\sum_{i=1}^{N_S}\sum_{j=1}^{N_S}\left[\frac{d_{ij}}{d_{max}\cdot n_{taxi}}(1-\gamma)\cdot x_{ij}^{t}\right] + \sum_{t=1}^{H}\sum_{i=1}^{N_S}\left[\frac{1}{n_{taxi}^{2}}\gamma\cdot(y_i^t)^2\right] \tag{2.11}$$

Therefore, the optimization problem solved at each rebalancing time step, and mentioned in Algorithm 2, is expressed as follows:

$$\min_{x,y,z} \quad \sum_{t=0}^{H}\sum_{i=1}^{N_S}\sum_{j=1}^{N_S}\left[\frac{d_{ij}}{d_{max}\cdot n_{taxi}}(1-\gamma)\cdot x_{ij}^{t}\right] + \sum_{t=1}^{H}\sum_{i=1}^{N_S}\left[\frac{1}{n_{taxi}^{2}}\gamma\cdot(y_i^t)^2\right]$$

subject to

- $x, z \in \mathbb{N}$
- $y \in \mathbb{R}$
- $y_i^0 = \phi_{i_{source}}^0 - \sigma_i^0 + z_i^0 + x_{i_{out}}^0 \qquad \text{for } i = 1,...,N_S$
- $y_i^t = y_i^{t-1} - \sigma_i^t + x_{i_{out}}^t - x_{i_{in}}^t - z_{i_{end}}^t + z_i^t + \phi_{i_{source}}^t - \phi_{i_{dest}}^{t-path}$ $\qquad \text{for } t = 1,...,H \text{ and } i = 1,...,N_S$
- $x_{i_{out}}^0 + z_i^0 \leq \sigma_i^0 \qquad \text{for } i = 1,...,N_S$
- $x_{i_{out}}^t + z_i^t \leq \sum_{\tau=0}^{t-1}(\sigma_i^\tau + x_{i_{in}}^\tau + z_{i_{end}}^\tau + \phi_{i_{dest}}^\tau - x_{i_{out}}^\tau - z_i^\tau) +$ $+\sigma_i^t + x_{i_{in}}^t + z_{i_{end}}^t + \phi_{i_{dest}}^t \qquad \text{for } t = 1,...,H \text{ and } i = 1,...,N_S$

(2.12)

The method results then in a predictive optimization problem, thanks to the inclusion of the predictions of future data in it. This leads to obtaining at each time step a solution that aims at minimizing the overall cost of the simulation, and not only the present cost.

3

Implementation

This chapter describes the implementation of the algorithm described in Section 2.4 on a small sample network in a python environment, followed by the analysis of its results. Subsequently, the application of the same theory on a real network implemented of the software AMoDeus is reported. The chapter then concludes with the analysis of the results of the real network simulation. As described in the preface, the sample network implementation (Section 3.1) was primarily done by an other student in the team, so it is briefly summarized here.

3.1 Sample network - python

A small-scale system was first implemented in python in order to test and optimize the algorithm. The network was modelled as a graph composed of 9 stations, as shown in Figure 3.1. The roads that connect the different stations are represented through the indirect edges of the graph that can be run in both ways.

The demand was simulated by sampling a Poisson distribution generated with different expected rates of occurrences for each station and time of the day. In the simulation the time is discrete, so all the data needed to be divided in time step. The demand corresponding to a full day, evaluated as previously explained, was stored in 180 time steps, meaning that each time step corresponds to a time interval of 8 minutes.

For what concerns the charging, every time that a taxi travels from one node to another, its SoC is reduced of an amount equal to the weight of the edge that connects the two nodes. When the vehicle is plugged to the charging station, instead, its battery increases of a value Δ_{soc} every time step.

In the beginning of each time step, the static assignment (see Section 2.4.1) takes place, involving all the customers that happen to be in the same location of an available taxi whose state of charge is high enough to complete the customer's trip. The next step can then take place: a cost matrix is built as summarized in Algorithm 1, and processed with the Hungarian method in order to find the optimal dispatching decision for the customers that are still waiting. Here taxis can be reactively unplugged when needed.

In conclusion, the last step consists in the solution of the optimization problem (see

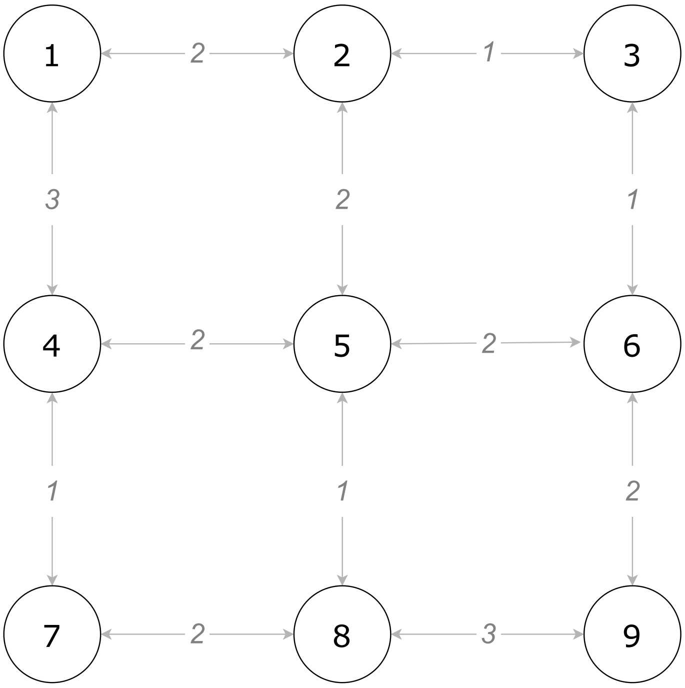


Figure 3.1: *Network implemented in python.*

Section 2.4.3), to obtain a series of rebalancing actions corresponding to each time step of the time horizon, of which only the first one will be applied.
The optimization problem was formulated and solved with the aid of the optimization software package CPLEX by IBM [30].

Actions implemented to improve scalability

The problem described through the cost function in the rebalancing step of the algorithm is a Mixed Integer Quadratic Programming (MIQP) problem. This means that, for large scale problems, the application of the method described might become unfeasible. To solve this issue, a relaxation on the integer constraint was adopted, as previously done in [22]: in the MPC, the values of all the decision variables for all the time steps of the horizon are computed, but then only the first step of the resulting strategy is implemented. For this reason, only the decision variables x_i^0 and z_i^0 of the first time step actually need to be integer, in order for the algorithm to be applicable: the decision variables of the other steps are set to be of the continuous type instead. In this way, the dependence of the number of integer constraints in the problem from the length of the time horizon H is relaxed, thus making the method more scalable and potentially applicable to larger systems or longer horizons, while the risk to lose some optimality in the solution is minor.

EXAMPLE 1 - dispatching

Here is a short example to visually understand how the dispatching phase of the algorithm works.

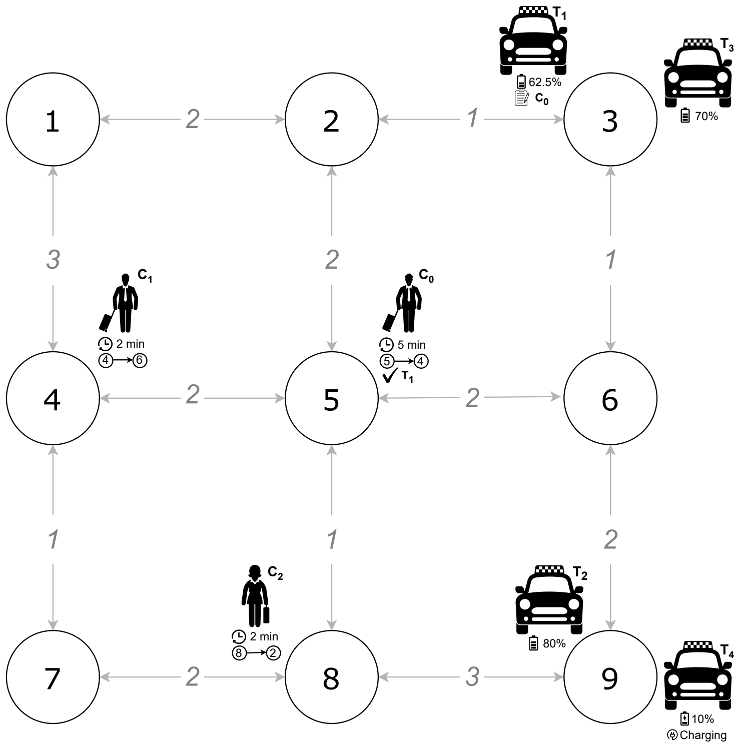


Figure 3.2: *Scenario of dispatching example.*

The initial scenario is the one showed in Figure 3.2, there are four taxis in the network: T_1, T_2, T_3, T_4. Taxi T_1, whose battery level is 62.5%, is currently in station 3, but it is travelling to station 5 to pick up customer C_0, whose destination is station 4. Taxi T_3, with a SoC of 70% is in the same station and it is not moving, while in station 9 there are taxis T_2 (SoC = 80%) and T_4 (SoC = 10%), which is charging. Besides C_0, that has already been assigned previously and is only waiting to be picked up, there is also customer C_1, in station 4 with destination 6, and C_2, in station 8 with destination 2. Both customers arrived just 2 minutes ago (Waiting Time = 2).

The following calculations show how the elements cost matrix for the solution of the dispatching problem are computed.

Customer $\mathbf{C_1}$:

- $\mathbf{T_1} \rightarrow c_{11} = \frac{P_{11}+P_{res_1}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_1}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_1}{MAX_{SoC}} \cdot w_{SoC} =$
  $= \frac{0+5}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-62.5}{100} \cdot 0.1 = 0.444$

- $\mathbf{T_2} \rightarrow c_{21} = \frac{P_{21}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_1}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_2}{MAX_{SoC}} \cdot w_{SoC} + c_s \cdot w_s =$
  $= \frac{6}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-80}{100} \cdot 0.1 + 1 \cdot 0.1 = 0.608$

- $\mathbf{T_3} \rightarrow c_{31} = \frac{P_{31}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_1}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_3}{MAX_{SoC}} \cdot w_{SoC} + c_s \cdot w_s =$
  $= \frac{5}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-70}{100} \cdot 0.1 + 1 \cdot 0.1 = 0.536$

- $\mathbf{T_4} \rightarrow c_{41} = \infty$ (taxi T_4 does not have enough battery to reach customer C_1)

Customer $\mathbf{C_2}$:

- $\mathbf{T_1} \rightarrow c_{12} = \frac{P_{12}+P_{res_i}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_2}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_1}{MAX_{SoC}} \cdot w_{SoC} =$
  $= \frac{5+3}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-62.5}{100} \cdot 0.1 = 0.688$

- $\mathbf{T_2} \rightarrow c_{22} = \frac{P_{22}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_2}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_2}{MAX_{SoC}} \cdot w_{SoC} + c_s \cdot w_s =$
  $= \frac{3}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-80}{100} \cdot 0.1 + 1 \cdot 0.1 = 0.364$

- $\mathbf{T_3} \rightarrow c_{32} = \frac{P_{32}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_2}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_3}{MAX_{SoC}} \cdot w_{SoC} + c_s \cdot w_s =$
  $= \frac{4}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-70}{100} \cdot 0.1 + 1 \cdot 0.1 = 0.455$

- $\mathbf{T_4} \rightarrow c_{42} = \frac{P_{42}}{MAX_P} \cdot w_P + \frac{MAX_{WT}-WT_2}{MAX_{WT}} \cdot w_{WT} + \frac{MAX_{SoC}-SoC_4}{MAX_{SoC}} \cdot w_{SoC} + c_s \cdot w_s =$
  $= \frac{3}{8} \cdot 0.65 + \frac{2-2}{2} \cdot 0.15 + \frac{100-10}{100} \cdot 0.1 + 1 \cdot 0.1 = 0.434$

The cost matrix for the Hungarian algorithm is showed in Table 3.1:

	C_1	C_2
T_1	0.444	0.688
T_2	0.608	0.364
T_3	0.536	0.455
T_4	∞	0.434

Table 3.1: *Cost Matrix of Example 1.*

In this case the steps of the Hungarian algorithm will be neglected, in fact optimal assignment results trivial: $\begin{cases} T_1 \rightarrow C_1 \\ T_2 \rightarrow C_2 \end{cases}$

The preferred assignment for customer C_1 results to be with T_1, because that taxi already has, in its planning, the task to go to station 4, where C_1 is situated. For this reason it is preferred over taxi T_3 that is now in its exact same position, even though it has a higher SoC. This happens thanks to the contribution of the starting cost: taxi T_1 is going to pick up customer C_0 and will transport it to station 4, so it will take in total 5 time steps for T_1 to be ready to pick up C_1. If T_3 was chosen for C_1, 5 time steps would still pass before the pick up can be done. The energy consumption connected to the trip of taxi T_3 to reach station 4 is then completely unnecessary and can be avoided thanks to the contribution of c_S. Taxi T_2 is not chosen for C_1 because it is more distant from it than T_1 and, once again, it has also the contribution of the starting cost; while taxi T_4 does not have enough battery to pick up C_1. For customer C_2 instead taxi T_2 is chosen, because it is in the same station of T_4 but has a higher battery level, and it is closer than T_1 and T_2.

EXAMPLE 2 - rebalancing

The following example shows how the decision-making process of the rebalancer works, in the very simple scenario of Figure 3.3.

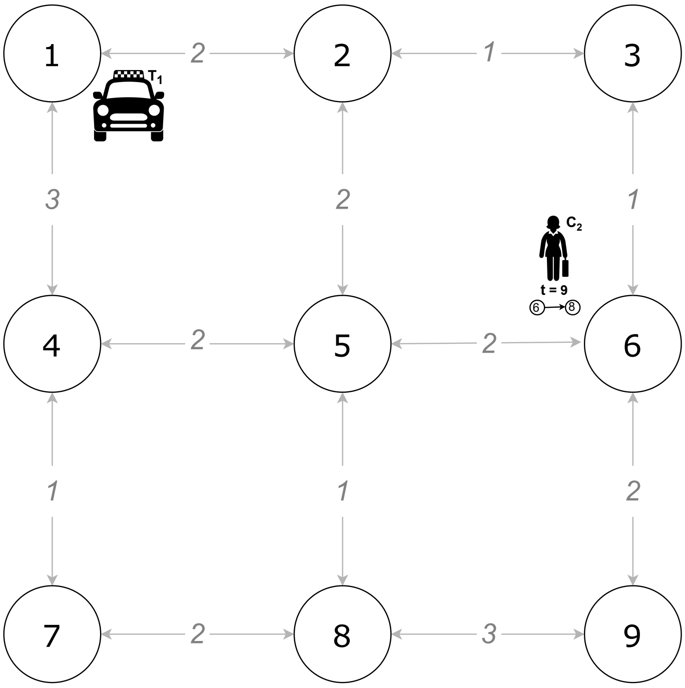


Figure 3.3: *Scenario of rebalancing example.*

The horizon used is equal to 5 time steps, the taxi is waiting in station 1 while the customer will appear at time step 9 in station 6 with station 8 as destination.

***t* = 4)** The customer appears in the last time step of the horizon $t_h = 5$. The shortest path from station 1 to station 6 takes 4 time steps, so if the taxi started moving now then it would arrive before the customer appears. Thus, no rebalancing action is made.

***t* = 5)** At this time step $\phi^4_{6_{source}} = 1$ meaning that the forecaster predicted that a customer will appear in station 6 at time step of the horizon $t_h = 4$. This leads to the chain of decisions showed below, where red text highlights the decision taken from the rebalancer to minimize the cost - omitted terms are equal to 0:

- $y^4_6 = \phi^4_{6_{source}} - x^4_{6_{in}}$: to have $y^4_6 = 0 \rightarrow x^4_{6_{in}} = 1$
- to have $x^4_{6_{in}} = 1 \rightarrow x^3_{36} = 1$

- $x^3_{3_{out}} = x^3_{36} + x^3_{32} = 1 + 0 = 1$
- $y^3_3 = -x^3_{3_{in}} + x^3_{3_{out}}$: to have $y^3_3 = 0 \rightarrow x^3_{3_{in}} = 1$
- to have $x^3_{3_{in}} = 1 \rightarrow x^2_{23} = 1$
- $x^2_{2_{out}} = x^2_{23} + x^2_{25} + x^2_{21} = 1 + 0 + 0 = 1$
- $y^2_2 = -x^2_{2_{in}} + x^2_{2_{out}}$: to have $y^2_2 = 0 \rightarrow x^2_{2_{in}} = 1$
- to have $x^2_{2_{in}} = 1 \rightarrow x^0_{12} = 1$

The outcome of this chain is that $x^0_{12} = 1$, and according to the MPC policy the system executes only the first decision of the horizon: one taxi is then sent from station 1 to station 2.

***t* = 6)** Now, the taxi is on the way to reach station 2 from station 1 (the trip takes two time steps), and it is marked in the term σ^1_2, that counts the taxis that will arrive at time step $t + t_h = 6 + 1 = 7$ in the station 2.

- $y^3_6 = \phi^3_{6_{source}} - x^3_{6_{in}}$: to have $y^3_6 = 0 \rightarrow x^3_{6_{in}} = 1$
- to have $x^3_{6_{in}} = 1 \rightarrow x^2_{36} = 1$
- $x^2_{3_{out}} = x^2_{36} + x^2_{32} = 1 + 0 = 1$
- $y^2_3 = -x^2_{2_{in}} + x^2_{2_{out}}$: to have $y^2_3 = 0 \rightarrow x^2_{2_{in}} = 1$
- to have $x^2_{3_{in}} = 1 \rightarrow x^1_{23} = 1$
- $x^1_{2_{out}} = x^1_{23} + x^1_{25} + x^1_{21} = 1 + 0 + 0 = 1$
- $y^2_1 = \sigma^1_2 - x^1_{2_{in}} = 1 - 1 = 0$

The evaluation follows the same phases of the one in the previous time step, however it ends without any physical action because the taxi sent in the previous time step is already on the way to station 6.

***t* = 7)** Afterwards, when the taxi arrive in station 2, the evaluation ends assigning the taxi to the following rebalancing trip, heading to the station 3:

- $y^2_6 = \phi^2_{6_{source}} - x^2_{6_{in}}$: to have $y^2_6 = 0 \rightarrow x^2_{6_{in}} = 1$
- to have $x^2_{6_{in}} = 1 \rightarrow x^1_{36} = 1$
- $x^1_{3_{out}} = x^1_{36} + x^1_{32} = 1 + 0 = 1$
- $y^1_3 = -x^1_{3_{in}} + x^1_{3_{out}}$: to have $y^1_3 = 0 \rightarrow x^1_{3_{in}} = 1$
- to have $x^1_{3_{in}} = 1 \rightarrow x^0_{23} = 1$

$\boldsymbol{t = 8}$**)** Finally, in the time step just before the appearance of the customer, the rebalancer moves the taxi to station 6, so that it will be available to pick him/her up:

- $y_6^1 = \phi_{6_{source}}^1 - x_{6_{in}}^1$: to have $y_6^1 = 0 \rightarrow x_{6_{in}}^1 = 1$
- to have $x_{6_{in}}^1 = 1 \rightarrow x_{36}^0 = 1$

The method described in this example is also able to select the shortest path among the network. In fact, for each action that the rebalancer issues, the cost of the optimization problem increases proportionally to the length of the path (see Section 2.4.3). Thus, the path chosen is the one that in the first place allows to pick up the customer, and then increases the total cost as less as possible.

3.2 Theoretical Results

This section compares and analyzes different aspects and outcomes of the T-EAMoD algorithm applied to the sample network with the set-up described in Section 3.1.

Sensitivity analysis to the service parameter γ

In order to evaluate the optimal relative importance of the two cost function components, some analysis were performed by solving the optimization problem adopting different values of the parameter γ that varies from 0 to 1 on Equation 2.11.

First of all, a scenario with no demand was constructed, where all the taxis were situated in the central station at the beginning of the simulation. The results showed that, for values of the service parameter lower than 0.01, the taxis do not move at all: the cost of the imbalance in this case results almost null, while the expense connected to moving empty vehicles is very large. As γ increases, as long as $\gamma < 0.6$, the vehicles start spreading along the network, but not uniformly. In particular, for low values of γ, fewer taxis move, and only towards stations closer to their starting point, while as the service parameter gets higher the taxis start moving also towards the periphery of the network, and in general a higher number of vehicles moves. An equilibrium is always reached, but the amount of taxis among the different stations is different. A perfectly balanced equilibrium, with the same number of vehicles in each station, is reached only when $\gamma = 0.6$: in this case the overall cost results to be about 1. For values of the service parameter higher than 0.6, an equilibrium is never reached anymore and there is a continuous ping-pong effect of taxis among stations: a high imbalance penalizes the system more and more, while the cost of sending around empty vehicles always becomes smaller. The rebalancer tends to keep vehicles in movement because, when a taxi leaves a station and is on its way to another one, it will not contribute to the imbalance of the two in the time steps during which it is travelling. For this reason, the preferred station chosen for the rebalancing in this case is the farthest away, so that the taxi will have to do a longer journey and will not contribute to the imbalance of any station for longer. Moreover, as γ increases, clusters of taxis of growing dimensions will tend to form, which move together around the network until, in the worst case ($\gamma = 1$), all the taxis start moving together. This set of experiments proves that the rebalancing algorithm works as expected, as it always drives the system towards a minimization of the real overall price, even in the limit cases.

When the same experiments are performed on Equation 2.2 with values of $\widetilde{y_i}^t$ different from zero, the equilibrium configuration changes, but the system shows the same behavior. For values of γ lower than 0.6, it does not tend anymore to an even distribution of vehicles among the stations, but instead it aims at replicating the equilibrium described by $\widetilde{y_i}^t$, which is perfectly reached only for $\gamma = 0.6$.

It is necessary to point out that the previous analysis was done with a modified version of the cost function described in 2.2 or in 2.11, that includes the term y_i^0, related to the imbalance of time step 0. This was necessary in order to visualize the outcome of the decision-making process of the rebalancer which, in case of absence of demand, would not visible if the term y_i^0 was not included in the cost function. Without y_i^0, in fact, the simulation would result in a static behavior of the taxis, which would not move at all unless a $\gamma = 1$ is adopted. A deeper understanding of the real difference between these two options can be reached by investigating in detail the functioning of the MPC rebalancing algorithm. As explained in Section 2.4.3, in fact, at every time step there is an evaluation of the optimal control inputs for the whole time horizon. As a consequence of this, if the imbalance term y_i^0 is incorporated in the cost function, the output of the rebalancer will include actions referring to the present time (instant $t = 0$) which aim at reducing such value of present imbalance. Differently from the actions computed for the subsequent time steps of the horizon, which will eventually be re-evaluated, those referring to the present moment are actually put into action: the risk is that many vehicles might be moved just in order to see a net reduction of imbalance at the moment being, even when this motion is not justified by a real or predicted presence of customers somewhere in the network.
Removing y_i^0 from the optimization problem, as was actually done in this research, means instead that the movement of taxis by the rebalancer is driven by the presence of either real or forecasted customer demand. Some tests were conducted to prove these different behaviors of the system in case of absence and presence of the term y_i^0 from the cost function. The cases simulated included a fictitious demand generated with Poisson samples with varying values of λ for different stations and hours, as previously explained. The results in Figure 3.4 show that the behavior of the overall waiting time as a function of γ is quite similar in the two cases, both in terms of trend and values. On the other hand, from the analysis of the empty distance (Figure 3.5) it is possible to note that the presence of y_i^0 drives the system to an instability condition for values of $\gamma > 0.6$, thus confirming the behavior previously studied in absence of demand.

Subsequently, for the choice of the appropriate value of service parameter to use in the simulations, and for a better understanding of the physical meaning of the different values it can be given, the same scenario just described is simulated using

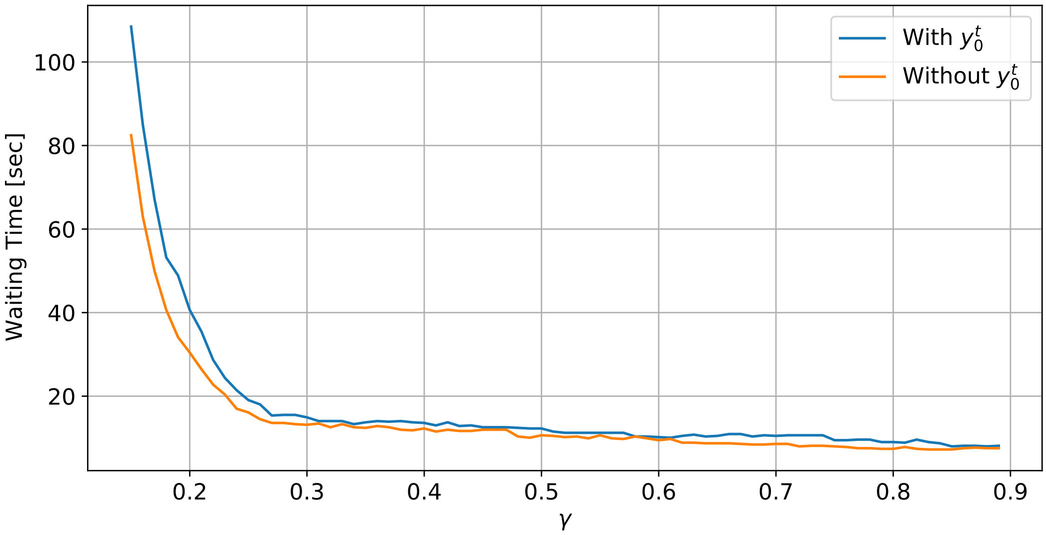


Figure 3.4: *Comparison between the overall waiting time obtained using the cost function with and without y_0^t. The times are evaluated as a function of the service parameter γ that varies from* 0.2 *to* 1 *with a horizon length of 15 time steps, simulated in the sample network.*

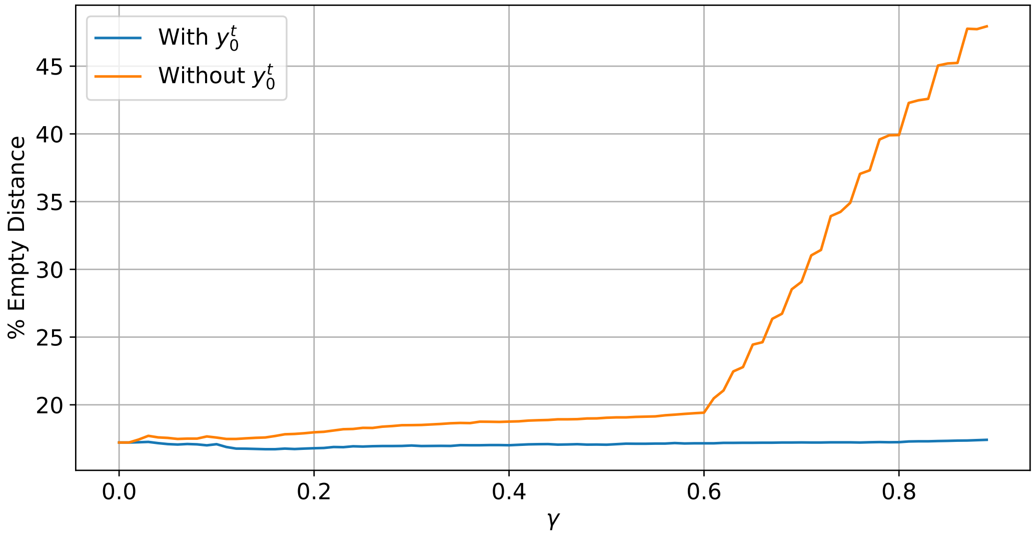


Figure 3.5: *Comparison between the percentage of empty distance obtained using the cost function with and without y_0^t. The distance is evaluated as a function of the service parameter γ that varies from* 0 *to* 0.85 *with a horizon length of 15 time steps, simulated in the sample network.*

the classical cost function described in 2.11, i.e. without the y_i^0 term, for varying values of γ. In particular, the tests conducted showed the influence the parameter γ has on the total customers' waiting time and on the overall empty distance travelled by the vehicles. The case simulated had all the taxis starting in the central station, and a fictitious demand was generated with Poisson samples with varying values of λ for different stations and hours, as explained before. In the rebalancing step, the known future demand was used (oracle), and a horizon of 15 time steps was adopted, corresponding to a time of two hours. The results in Figure 3.6 show that the curves of the waiting time and empty distance percentage have inverse behaviors: in order

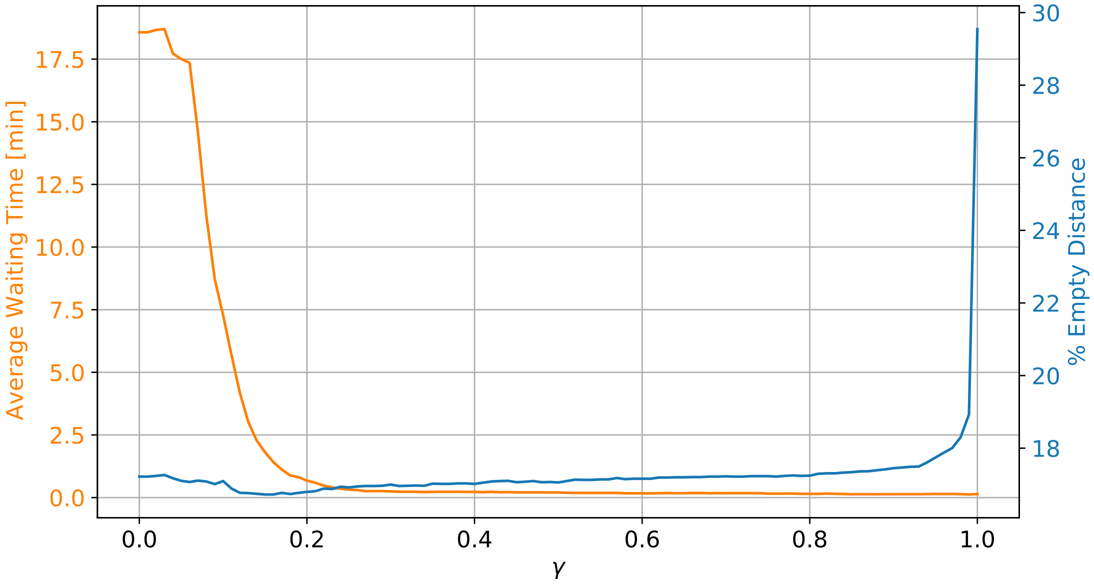


Figure 3.6: *Waiting time and percentage of empty distance as a function of the service parameter γ with a horizon length of 15 time steps, simulated in the sample network.*

to design an AMoD system which is as efficient as possible, without compromising the quality of the service it provides, a trade-off between the two objectives must be done. To explain the behavior of the empty distance percentage curve shown in Figure 3.8 it is necessary to go back to the physical meaning of the parameter γ. Adopting a $\gamma = 0$ would lead to not do any rebalancing at all. In fact in that case the imbalance would have no influence over the value of the cost function, while the cost of rebalancing vehicles would be very high: the optimal solution would then be to not move empty taxis. As the weight of the imbalance term on the overall cost increases, the controller starts issuing some rebalancing actions, that have the positive influence of reducing the distance travelled by the vehicles to pick up customers. Until the reduction of pick up distance is bigger than the increase in rebalancing distance, the curve of the empty distance percentage shows a deflection. This trend sees an inversion when the rebalancing trips become more onerous, and the increase in energy consumption they bring is higher than the savings they cause by reducing the pick up trips. Looking at Figure 3.7, it is possible to note that low values of the service parameter cause a steep increase in customers' waiting time, that grows uncontrollably when γ decreases below 0.3. For this reason, the values of gamma between 0.16 and 0.19 that, according to Figure 3.8, would lead to a reduction in empty distance percentage, cannot be adopted. A good trade-off between the two objectives would be to select $\gamma = 0.7$, which shows that the distance travelled by empty vehicles is very close to the situation without rebalancing, thus bringing good benefits in terms of waiting time reduction.

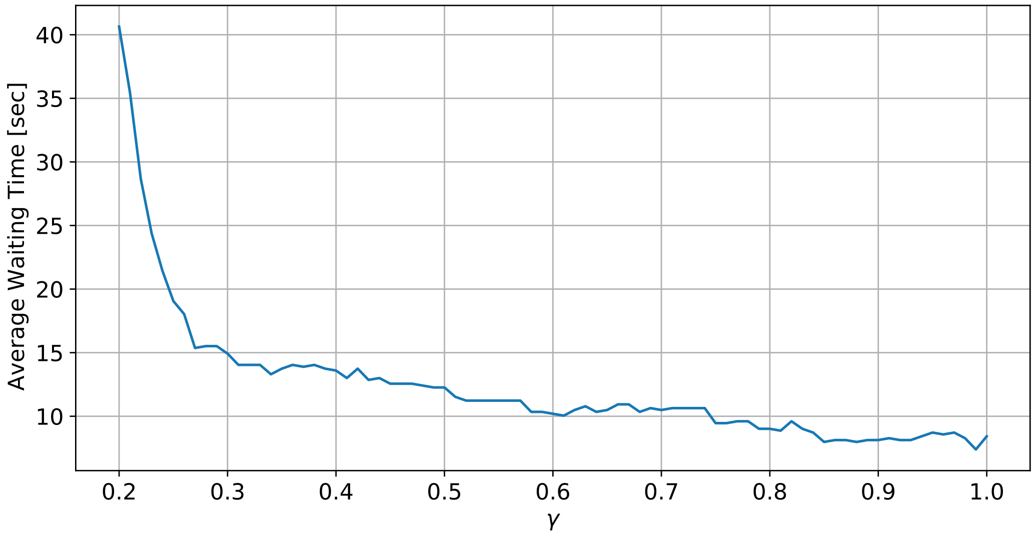


Figure 3.7: *Total waiting time as a function of the service parameter γ that varies from* 0.2 *to* 1 *with a horizon length of 15 time steps, simulated in the sample network.*

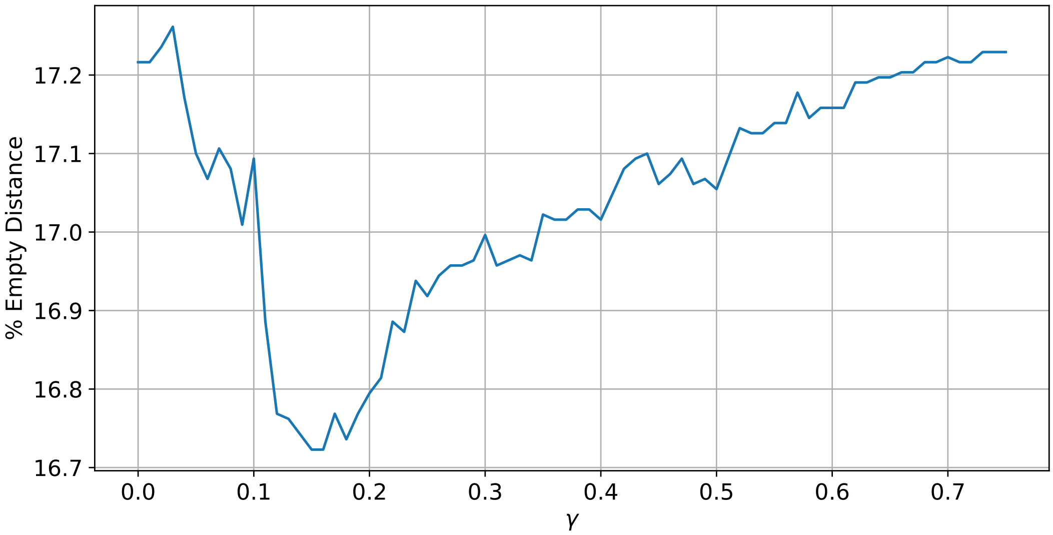


Figure 3.8: *Percentage of the empty distance as a function of the service parameter γ that varies from* 0 *to* 0.95 *with a horizon length of 15 time steps, simulated in the sample network.*

Discussion about the optimal dimensions of the system

Considering a fixed total amount of customers along a whole day in all the simulations, an analysis was done in order to find the optimal size of the AMoD system in terms of number of vehicles and size of their battery. To this aim, the impact of different customers per taxi ratios (CT), defined as $n_{customer}/n_{taxi}$, on the performance of the system was investigated, repeating the simulations for different battery sizes. Being the system adopted here completely fictitious, the battery size is addressed in terms of amount of time steps that it allows to spend travelling, when fully charged. Some 3D graphs were first created to observe the behavior of the system in terms of waiting time and empty distance for different values of the two parameters, shown in Figures 3.9 and 3.10.

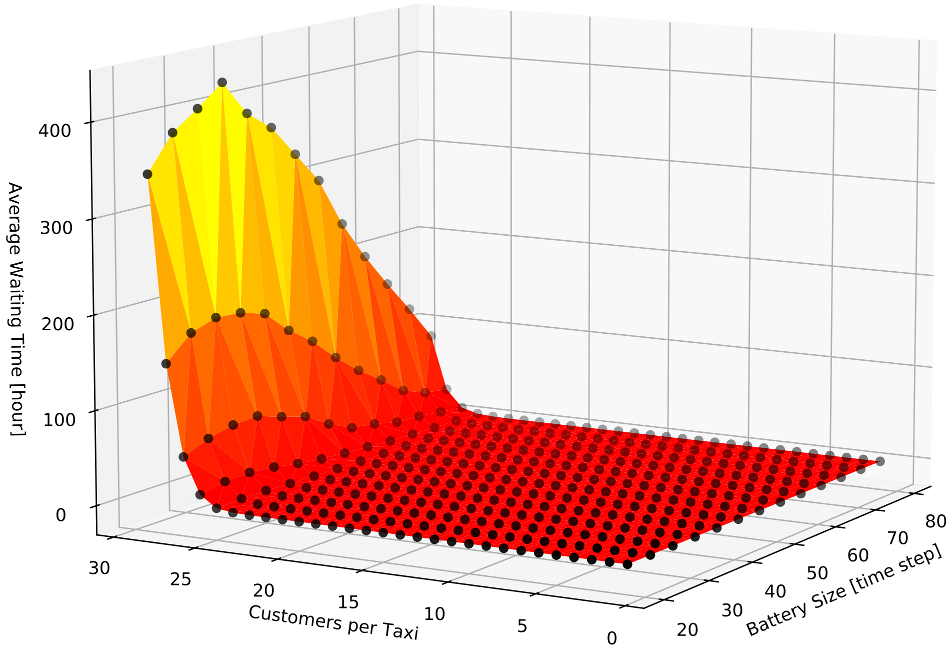


Figure 3.9: *Waiting time as a function of CT and of the battery sizes, simulated on the sample network with a horizon length of 15 time steps.*

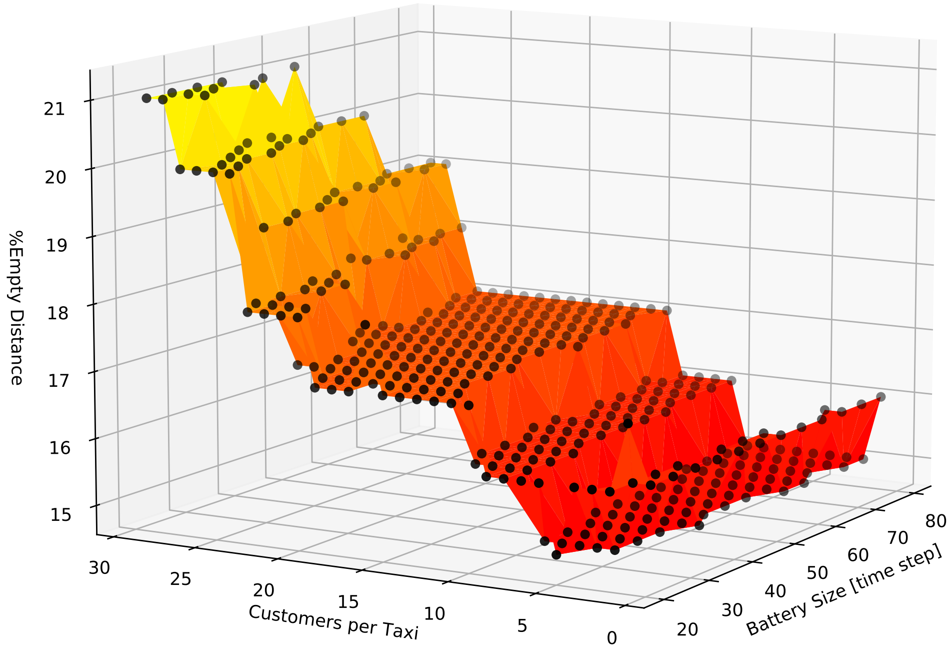


Figure 3.10: *Empty distance percentage as a function of CT and of the battery sizes simulated on the sample network with a horizon length of 15 time steps.*

To select the optimal CT and battery size it was necessary to keep in mind that having a big amount of vehicles, and/or equipping them with large batteries, causes the cost of the AMoD system to substantially increase.
A battery size of 40 time steps, in this system, corresponds to a travel time of about 5 hours and half, which is a reasonable value for electric cars nowadays. This value, assumed as a reference, was compared to a battery size of 20, its half, and of 80, its double: slices of 3D graphs were cut in correspondence of those values. The graphs obtained are reported in Figures 3.11 and 3.12.
For what concerns the waiting time, in Figure 3.11, its value remains within an acceptable range until CT$=23$, for a battery size of 20, or CT$=24$ if the dimension of the battery is 40. Adopting larger batteries it is possible to choose an even higher value of CT. Observing then Figure 3.12, it is possible to note that the curves corresponding to the three different battery sizes have the same trend, but the one representing a battery of 20 is always above the others of some decimal percentage points, thus resulting not convenient compared to them. Considering also the fact that it would not be realistic for this application, it was excluded. Since the other two curves are really close to each other, the battery size can be fixed at 40, which results a more economical choice than 80. These constraints the Customers per Taxi ratio to be fixed around $23-24$, as supported by both graphs. The values selected in this way were used for all the successive simulations performed onto the sample network.

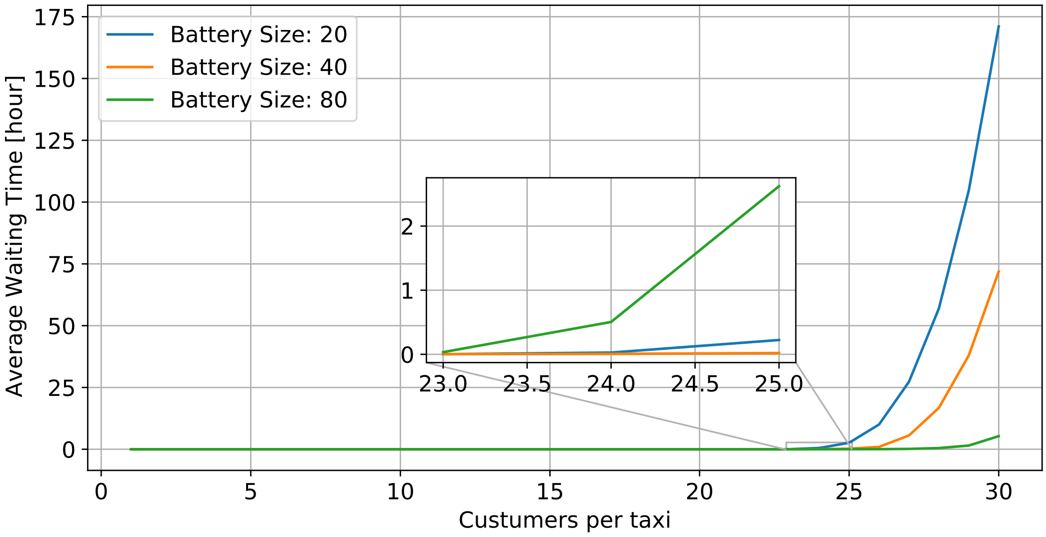


Figure 3.11: *Waiting time as a function of CT with battery sizes of* $20, 40, 80$ *simulated on the sample network with a horizon length of 15 time steps.*

Discussion about horizon

The length of the time horizon H adopted for the MPC in the rebalancing problem is a crucial factor for the quality of the results obtained, and its optimal value needs to be estimated carefully. Its lower bound is defined by the amount of time steps needed to travel between the two stations of the network that are the farthest to each

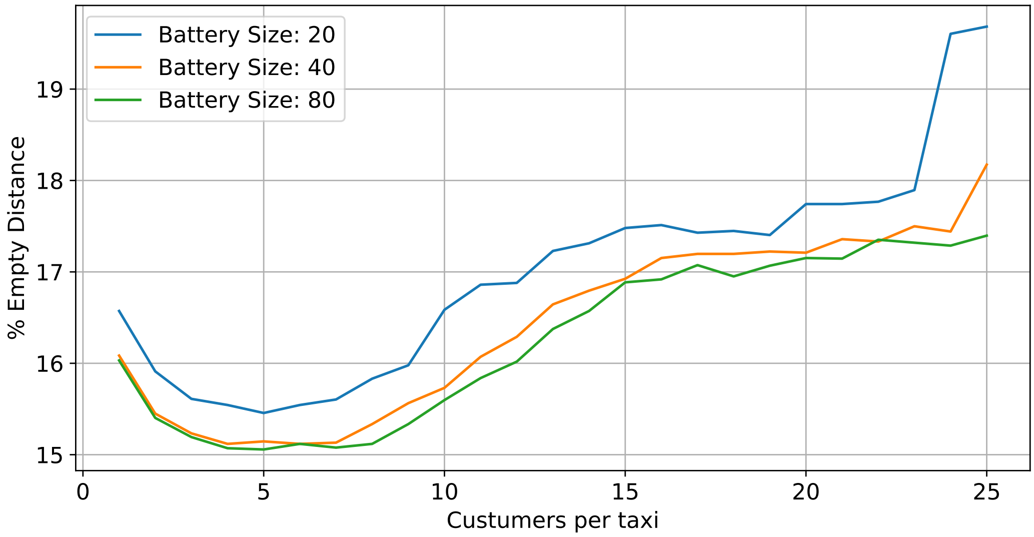


Figure 3.12: *Empty distance percentage as a function of CT with battery sizes of* $20, 40, 80$ *simulated on the sample network with a horizon length of 15 time steps.*

other, in order to ensure that, if a taxi is sent to rebalance in the present time step, its arrival in its destination happens before the end of the horizon. In fact, in case the horizon is not long enough, the risk is that the controller might not rebalance a taxi because it does not foresee the vehicle's arrival in its destination, and thus its action of customer pickup. In that case, the potential rebalancing action would just seem to cause an increase of cost without any positive outcome, so it would not be issued at all. This behaviour of the system, already explained in Example 2 of Section 3.1, is a consequence of the choice to allow the rebalancing of taxis only between adjacent stations. The benefit of this approach is to rebalance taxis step by step, from a station to another, avoiding to assign vehicles long rebalancing trips that would keep them out of the availability for long time. Although this method may increase the complexity of the system, it is mostly useful in applications where a predicted demand is used. Forecasted data are not completely reliable, so the fact that the rebalancer can move taxis only to neighboring stations gives it the ability of re-evaluating its decision at every time step. The forecast of data corresponding to a specific time step of the horizon, in fact, becomes more and more accurate as the time in which it is evaluated gets closer to the instant it refers to. In this way, if the forecaster changes its prediction with time, a taxi can still be sent back or re-directed somewhere else. On the other hand, if the forecast previously done is confirmed, then the taxi is already going in the right direction and it will continue following it.

Even though a real upper bound to the horizon length seems to not exist, for its selection it is necessary to take into account how the computational time of the simulation grows as it increases. As shown in Figure 3.13, in fact, the time needed to run each single time step of the simulation, with the setup adopted in this research[1],

[1] Setup used for the simulations: Dell Inspiron 7559 with Intel i7-6700HQ @ 2.60GHz and 16GB of RAM [31]. Software used: Jupyterlab with Python 3 and CPLEX 20.1.0 as optimizer.

increases significantly when H is extended.

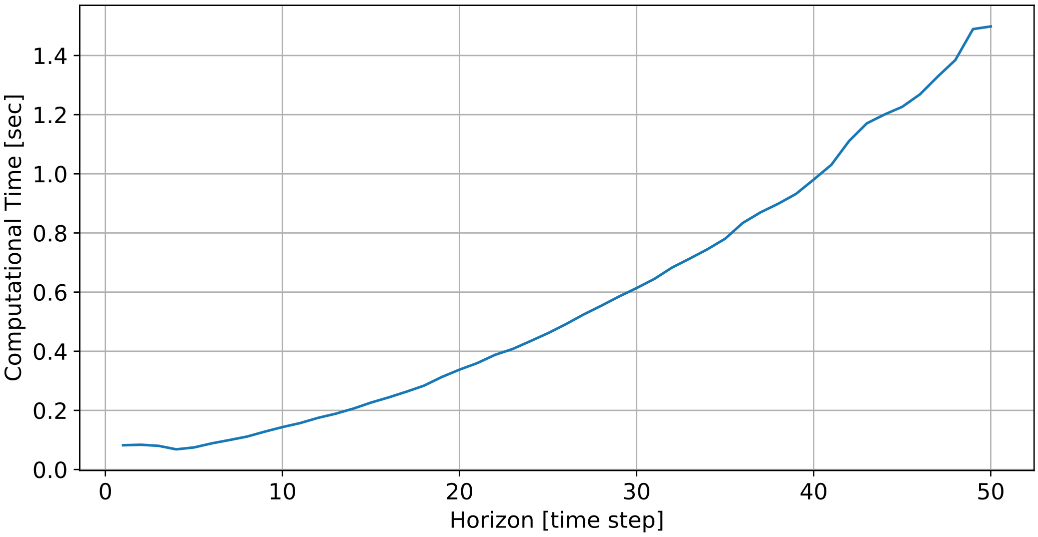


Figure 3.13: *Average of the computational time per time step as a function of the horizon length with $\gamma = 0.7$, simulated in the sample network.*

Looking at Figures 3.14 and 3.15, it is then possible to select the optimal length of the horizon time basing on the performance results in terms of customers waiting time and vehicles empty distance in function of H. Values of H below 8 are not acceptable from the point of view of the waiting time, because they cause a steep increase in that parameter. The maximum distance between any two stations in the network, which, as previously explained constitutes the lower bound to the length of the horizon, is in fact here of 8 time steps. In particular it is possible to observe a peak in the waiting time for a length of H of 2 time steps. Such a short horizon in fact limits the capability of the rebalancer to visualize all the possible stations in which each empty vehicle could be sent: only those at a short distance from the taxis are taken into account. This leads to a non-optimized solution whose results are even worse than the case with a shorter horizon. For $H = 1$, in fact, almost no action is issued at all, and this paradoxically results to be a better policy than executing the rebalancing without a proper horizon length.

For values of $H > 10$ instead the overall waiting time of the whole simulation seems to settle around acceptable values. A similar behavior can be observed when looking at the distance that vehicles without customers have to travel around the network to rebalance or to pick up customers. In this case, even though the variability of the parameter with H is very limited, it seems to almost reach a steady-state equilibrium when H is between 10 and 15. As a result of this analysis, an horizon length of 15 time steps, corresponding to a time of 2 hours, seemed a reasonable choice to be adopted in these simulations.

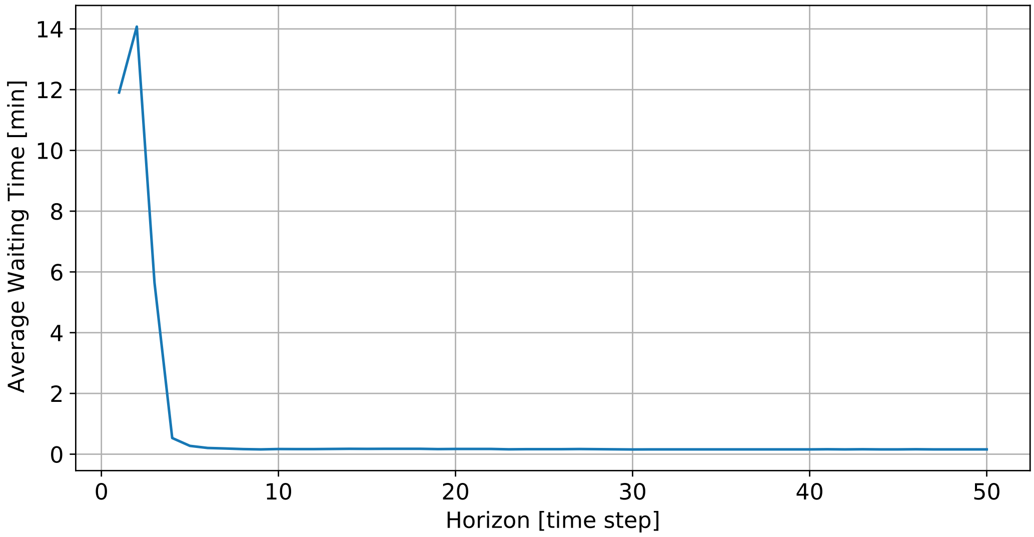


Figure 3.14: *Waiting time as a function of the horizon length with $\gamma = 0.7$, simulated in the sample network.*

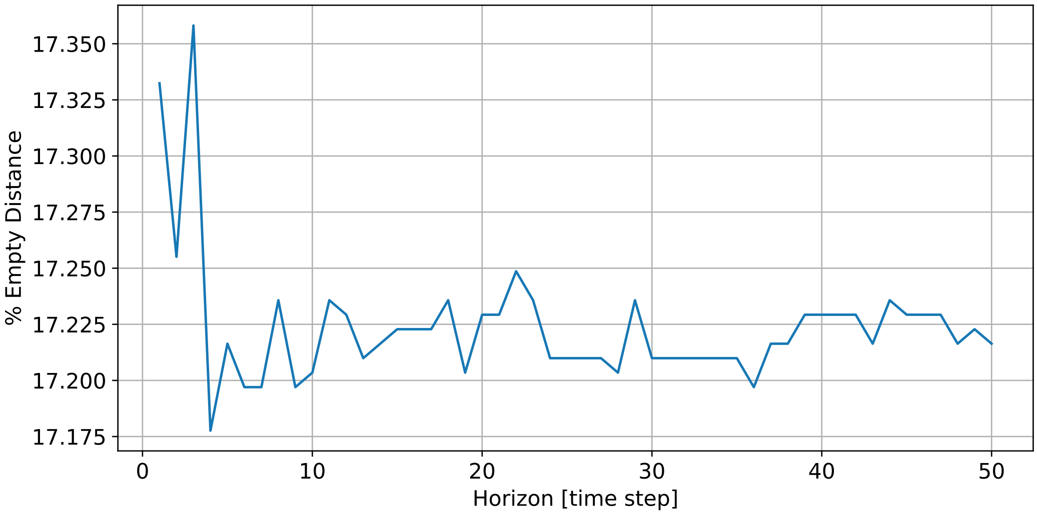


Figure 3.15: *Percentage of empty distance as a function of the horizon length with $\gamma = 0.7$, simulated in the sample network.*

Smart charging

As previously explained, the charging aspect was introduced into the optimization problem in order to optimize it while evaluating the rebalancing solution (smart charging). The variable z does not have a direct effect on the value of the cost function due to its weight W_z being set to zero, so its impact on the system comes from its presence in the constraint equations. The necessity to include it in the optimization problem comes from the fact that, if the taxis all start with full battery and remain in service until they have a sufficient SoC, they are likely to all end up needing charging at the same time, and often this happens right in correspondence of the peak hours of the demand.

The inclusion of the terms related to the charging in the constraints of the rebalanc-

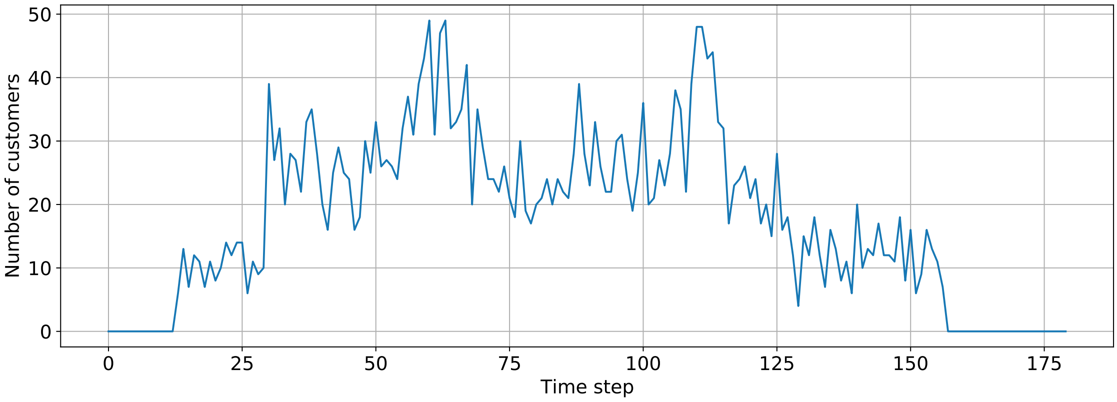


Figure 3.16: *Total number of requests per time step of the simulation.*

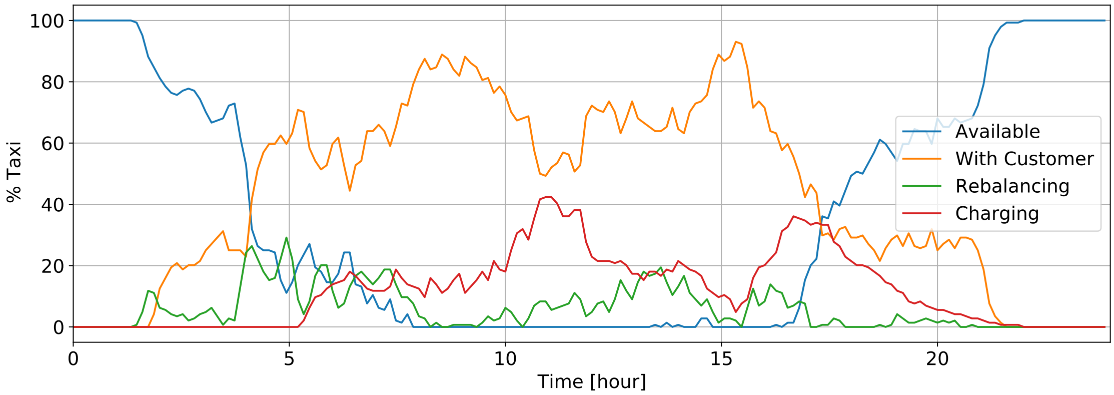


Figure 3.17: *Evolution of the state of the taxis for each time step in the case* ***without*** *the smart charging, simulated on the sample network with a horizon length of 15 time steps and* $\gamma = 0.7$.

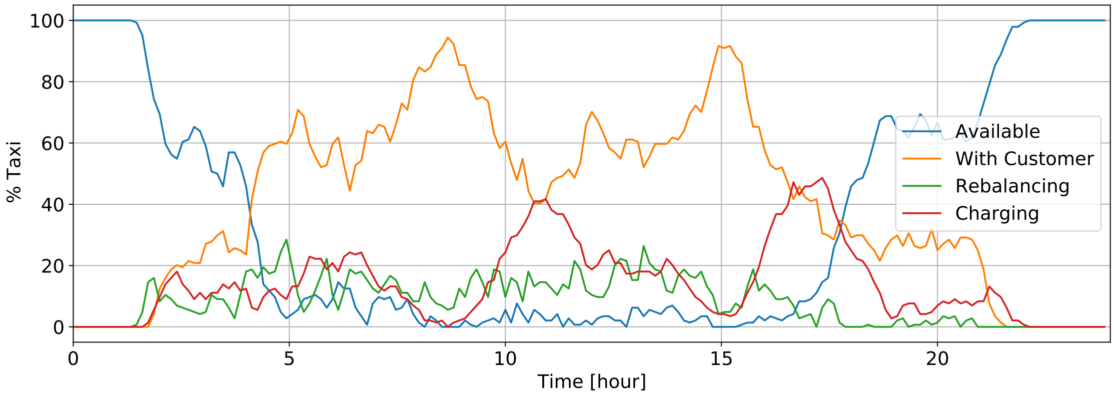


Figure 3.18: *Evolution of the state of the taxis for each time step in the case* ***with*** *the smart charging, simulated on the sample network with a horizon length of 15 time steps and* $\gamma = 0.7$.

ing has the effect of decreasing the peaks of the charging and spreading them along the day. Figures 3.17 and 3.18 show the evolution of the taxis states along the whole simulation done with the demand in Figure 3.16. By comparing these two it is possible to notice that with smart charging (3.18) the taxis start being plugged much earlier, and when the amount of customers in the network is at its peak most of the taxis have already charged and can thus sustain the elevate amount of requests. When the demand starts decreasing again, there is a new increase in the number of taxis that are plugged to charging stations. For the whole simulation, in the case where smart charging is adopted, the charging and occupied taxis continue having inverse behaviors, as can be seen by their curves that are almost perfectly mirrored.

When only a reactive charging is adopted, instead, taxis are plugged only when they need it. In this way, most of them end up needing battery during the peaks of the demand, which cannot then be properly satisfied, as testified by the fact that there is a saturation of available taxis and by the increase in customers' waiting time in Figure 3.19. This behavior is confirmed also by the trend of the vehicles' state of

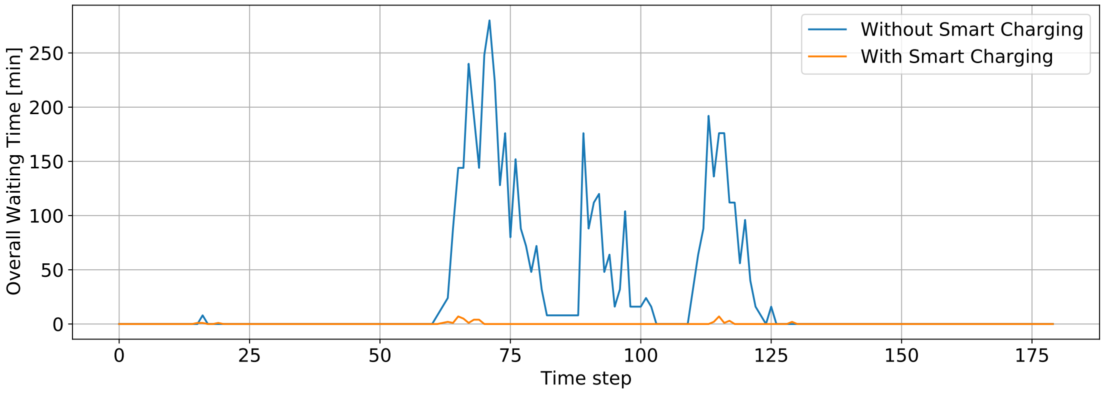


Figure 3.19: *Comparison of the overall waiting time resulting from the simulation of the sample network in the cases with and without smart charging, with a horizon length of 15 time steps and* $\gamma = 0.7$.

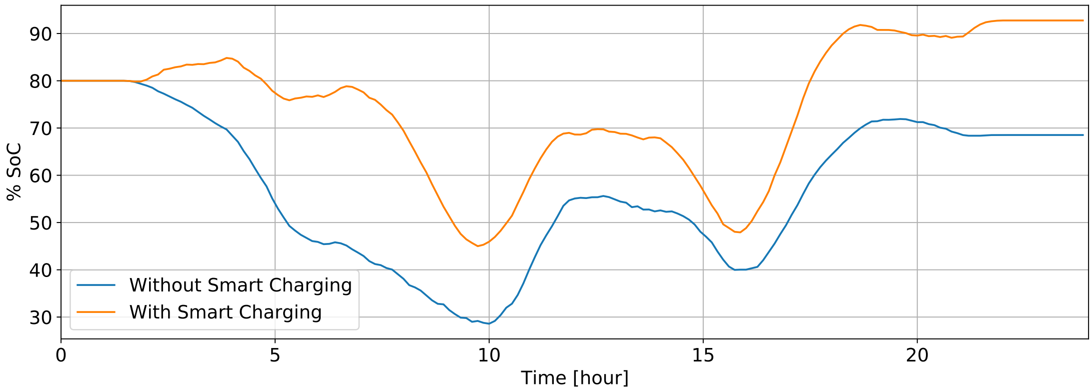


Figure 3.20: *Comparison of the average SoC of the fleet resulting from the simulation of the sample network with and without smart charging, with a horizon length of 15 time steps and* $\gamma = 0.7$.

charge, shown in Figure 3.20: all the vehicles start with a quite elevate SoC, but in the case with smart charging some of them start to charge very soon anyways. As a consequence, the drop in the overall level of battery in the case of smart charging is not as steep as it is in the case of reactive charging, and the average SoC of all the vehicles never goes below 45%, differently from the case with reactive charging, where it sometimes reaches vales lower than 30%. This behavior is an effect of the MPC approach adopted: when the rebalancer sees that not many taxis are needed at the moment but it knows, thanks to the predictions, that there will be an increase in the demand in the close future, it tries to immediately charge them, even if their battery is not at a minimum level, in order to have them available for when they will be more needed. On the other hand, in a moment when many vehicles are being requested by the customers, taxis will not be charged unless it is really necessary, because the rebalancer can foresee when the end of the demand peak will happen and knows whether they can wait for it or need to be charged before.

Comparison with the linear cost function

The necessity to account for the magnitude of the imbalance in the optimization problem, neglecting its sign, led to the possibility to define the cost function in two alternative ways. Equation 2.11, in fact, was compared to the following one, in which the squared term was replaced by an absolute value:

$$J_2 = \sum_{t=0}^{H} \sum_{i=1}^{N_S} \sum_{j=1}^{N_S} \left[\frac{d_{ij}}{d_{max} \cdot n_{taxi}} (1-\gamma) \cdot x_{ij}^t \right] + \sum_{t=1}^{H} \sum_{i=1}^{N_S} \left[\frac{1}{n_{taxi}} \gamma \cdot |y_i^t| \right] \tag{3.1}$$

Adopting equation 3.1 instead of equation 2.11 would heavily affect the optimization problem. In fact, in absence of the squared term, the problem would not be quadratic anymore, turning instead into a Mixed Integer Non Linear Programming (MINLP). The problem can be further simplified by rephrasing it as a MILP by removing the absolute value from y_i^t and replacing it with a secondary decision variable named $y_{abs_i}^t$, defined through the following constraints:

$$y_{abs_i}^t \geq y_i^t$$

$$y_{abs_i}^t \geq -y_i^t$$

$$y_{abs_i}^t \geq 0$$

As a result, an optimization problem of the complexity of a MILP is obtained.
The effects of the approaches with the original squared term (2.11) and with the modified version that include the absolute value (3.1) are evaluated in terms of customer waiting time and empty distance travelled, in function of the values of γ that can be adopted. The different ways the two functions impact the overall waiting time can be observed in Figure 3.21: the trends are quite similar, low values of γ correspond for both to a huge peak in waiting time, because in those cases rebalancing is almost not performed at all. The curve corresponding to the equation with the squared imbalance needs higher values of γ in order to settle to acceptable values of waiting time, but for $\gamma > 0.5$ the behaviors of the two are perfectly comparable.

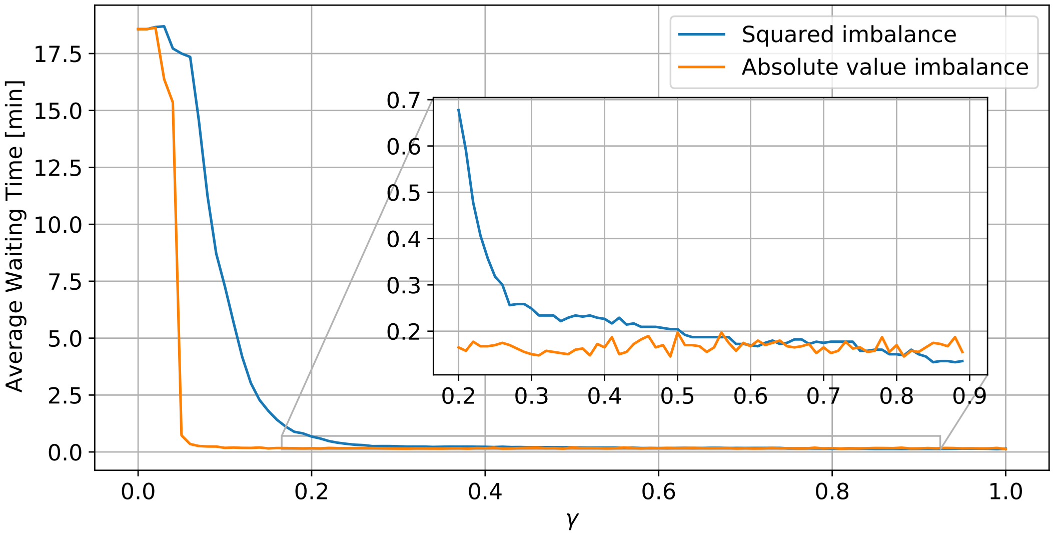


Figure 3.21: *Comparison between the waiting times obtained using the cost function with the squared imbalance and that with the absolute value of the imbalance. The times are evaluated as a function of the service parameter γ that varies from 0 to 1 with horizon a length of 15 time steps, simulated in the sample network*

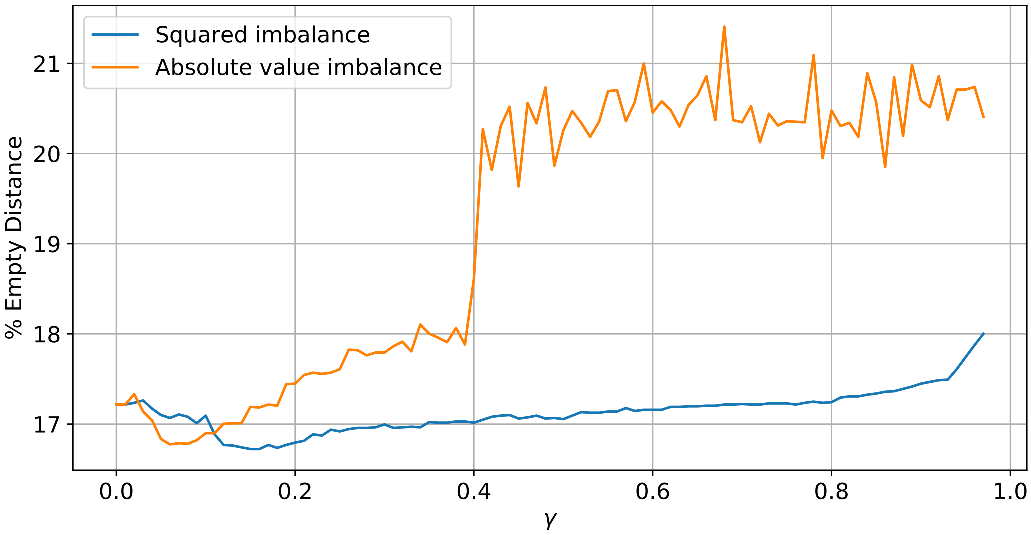


Figure 3.22: *Comparison between the percentage of empty distances obtained from the cost function with the squared imbalance and that with the absolute value of the imbalance. The distances are evaluated as a function of the service parameter γ that varies from 0 to 0.98 with a horizon length of 15 time steps, simulated in the sample network.*

For what concerns the empty distance travelled by vehicles, in Figure 3.22 it is possible to observe that in the case of cost function with absolute imbalance it sensibly grows for values of $\gamma > 0.4$, thus making it clear why the choice, in this research, fell on the squared imbalance version of the optimization problem. The odd behavior of the cost function in Equation 3.1 can be explained by noting that in its case, the imbalance in general weights less in the decision-making process, compared to the case where its term is squared. For this reason, when the rebalancer needs to select the proper vehicles to be sent in certain stations, the aspect of reducing the imbalance of other stations, when it is expressed as an absolute value and not a squared variable, is not a priority. In the case of Equation 2.11, instead, a criterion for the choice of the vehicle that needs to be rebalanced is also the amount of imbalance present in its starting station, and how much it can be reduced by sending such taxi away. This behavior in the case of Equation 3.1 leads to an accumulation of vehicles in the corners of the network, which results in longer empty trips due to a non-optimized behavior of the system.

3.3 Real network - AMoDeus

The algorithm created in the present research was then applied to the case study of the City of Chicago, with the aid of the open-source simulation environment AMoDeus (Autonomous Mobility on Demand Simulator) [32], which is an add-on to the multi-agent transportation simulator MATSim [33], implemented in Java. MATSim allows to simulate large transportation scenarios for one full day, modelling the traffic with a queue-based approach, leading to a maximization of the agents' (that here correspond to the customers) profit. AMoDeus has some built-in algorithms to test pre-written operational policies for mobility on demand systems, but it also allows to implement and test new policies developed by its users, so it resulted very useful for the purpose of this research.

Since the algorithm was written in python, it was necessary to establish a communication between the two applications through a socket, which works in the following way. At the beginning of each time step, AMoDeus sends information to the python script about: the customer requests appeared, comprehensive of their time of arrival, precise position and desired destination both in terms of coordinate and of node identification number, the taxis' positions and state, and the exact time. This information is then subjected to the previously explained operational policy, by processing it with the algorithm, and the commands that have to be adopted are then sent back to AMoDeus. Those correspond to indications about the taxi-customer matches and the rebalancing actions that have to be executed right away. When it receives information about an assignment, AMoDeus manages both the trip that the taxi needs to do to reach the customer's position before picking it up, if any, and the passenger's trip, unlike the python code where each part of trip needs to be separately accounted for.

The input to AMoDeus' simulations, the so-called scenario, should include also the city network, which was modelled as a directed graph in the python script. It involves all the information regarding each node, namely id and coordinates, and each link, which include: id, coordinates, nodes it is connected to, travel direction, nominal speed and length in meters. The city network for Chicago was already implemented in AMoDeus. In the network that describes Chicago city on AMoDeus, nodes not only represent crosses between roads, but are also placed along roads to subdivide each of them in smaller segments, resulting in more than 62 000 nodes and about 145 000 edges. For this reason, the probability of having a customer and a taxi in the same exact node is close to zero, thus the static assignment here was implemented in a slightly different way than how it is explained in Section 2.4. For each customer, the available taxi which is closer to it is first identified, using as metric the Euclidean distance. The taxi has to be inside a maximum radius of 100 meters for the customer in order to be taken into account. Its level of battery is checked, and if it results high enough to perform the whole customer's trip, the vehicle is immediately assigned to the person. Being the taxi really close to the customer, the time and space that separates them is negligible. In this way, a considerable number of customers, that were very close to available taxis, are assigned right away and do not need to enter the following phase.

The steps of the dispatching part of the algorithm here recall exactly those illustrated in Section 2.4.2. For what concerns the calculation the distance between two nodes, it was necessary to define whether the chosen path would need to be the shortest in terms of length travelled or time needed for it, which often do not correspond. If a totally realistic case was to be implemented, it would be necessary to account also for the congestion of the roads, that here is not considered. In that case, travel times would be strongly influenced by the traffic conditions, so it would be more meaningful to evaluate the distance between two points in terms of time needed to travel between them. Since this is not the case, and since the AMoDeus code, that manages the routing of the vehicles by itself, selects the taxis' routes according to their physical length, then the same method was adopted for the dispatching and rebalancing steps. This decision also reflects the main purpose of the present study: choosing to minimize the distance travelled by the taxis (in particular the empty distance they need to cover to reach customers), at the expense of the time needed for it, leads to better results in terms of energy efficiency. Since the network of Chicago provided by AMoDeus was given in terms of a directed graph, the algorithm used to compute distances along it is a bidirectional version of the Dijkstra shortest path [34].

In order to execute the rebalancing, the network needed to be partitioned to create the stations. A basic partitioning method was adopted, which consisted in dividing the city into rectangular areas and adjusting their dimensions according to data about the demand in different parts of the city, in order to obtain areas of different dimensions with similar amounts of customers during a whole day. The partition adopted can be seen in Figure 3.23. For each zone created by the partitioning process, a central node was defined: for most zones it corresponds to their geographic center, but in some cases it was positioned in a specific location with a particularly high density of demand. For example, the central node of zone 15 was placed in correspondence of Chicago airport. A graph was then created starting from this partition. Each zone is represented by its central node, while edges are defined to connect the neighboring stations. The weights of the edges depend on the distances between the central nodes of the two zones they connect. For each taxi or customer in the network, the coordinates of its position are used to find out the zone where it is located, and consequently the rebalancing node to which it belongs. It is necessary to point out that, differently from the sample network, in this case a customer's source and destination could also be in the same zone, which correspond to the same node in the graph: for this reason, self edges were added to the rebalancing graph. Their weights are defined as half of the diagonal of each zone. The optimization problem solved in this step works as expressed in Section 2.4.3. In this case, the interval between two consecutive rebalancing phases is 120 seconds, while the dispatching takes place every 10 seconds. In this way, the input data to the optimization problem can be gathered over a longer period of time, which makes them more meaningful. Moreover, being the rebalancing quite expensive in terms of computational time, reducing its frequency allows to speed up the whole simulation.

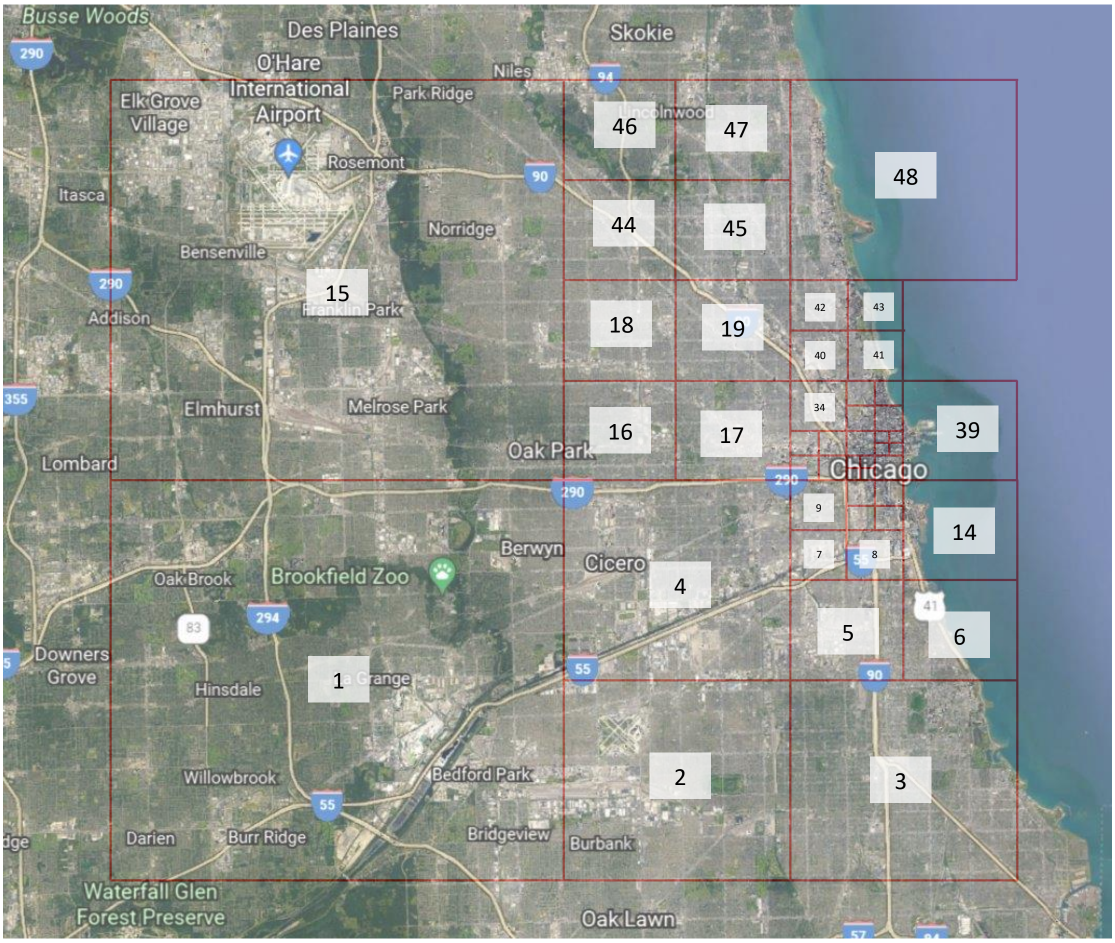


Figure 3.23: *Partitioned network for the rebalancing used in the AMoDeus simlations*

For what concerns the charging aspect, here some specific functions were implemented to account for it in a more realistic way. The real behavior of the curve of the state of charge of an electric vehicle depends on many different variables, some of which derive from the technical characteristics of the model of the vehicle, while others depend on the state of the road, such as its slope, congestion and speed limit, as well as on the atmospheric conditions and other factors. It would therefore be impossible to take them all in consideration in the context of this research: an accurate study regarding only the battery would be necessary. For this reason, charging and discharging of the taxis' SoC was approximated to be linear, and the data about the vehicle's battery capacity, duration and charging time were taken from those of the Volvo XC40 Recharge model [35]. The linearization of the charging functions is certainly an approximation, but it does not introduce large errors because the level of the battery in the simulations was mostly kept between 20% and 80%, and in that range the trend of electric vehicles' SoC can be considered almost linear.

Actions implemented to improve scalability

The use of the software AMoDeus allowed the simulation of a large-scale model, whose dimensions, amount of vehicles and passengers actually portrayed those of a real city. Although feasible, such simulation can become very expensive in terms of computational time, to the extent of being almost prohibitive if no corrective action is adopted to reduce it.

The role of the static assignment in this direction is crucial: when a taxi is really close to a customer, it is assigned right away, without having to pass by the dispatching step. In this way, the amount of customers on which the cost matrix for the Hungarian algorithm is built results much smaller.

There are two different computational time issues connected to the dispatching phase. The first one involves directly the time complexity of the Hungarian: as previously explained, it is $o(n^3)$, where n is the dimension of the matrix, corresponding to the largest between the number of taxis and of customers. It is clear, then, that reducing the size of the cost matrix sensibly decreases the computational time needed to process it and produce the dispatching decisions of each time step. The second reason why the dispatching phase becomes unfeasible in terms of computational time is instead related to the time complexity of the algorithm used for the evaluation of the shortest path. Considering that the running time of the Dijkstra's algorithm in the worst case performance is of the order $O(E + V \cdot \log(V))$, where E is the number of edges and V is the number of nodes [36], moving to a real-world dimension network penalizes this step of the algorithm and thus the whole duration of the simulation. An interesting option to solve this issue would be taking advantage of the partition created for the rebalancing to run the dispatching step within each of those stations, instead than in the whole network. However this action could lead to a reduction in the optimality of the dispatching solution, in fact vehicles would be assigned only to customers inside their same zone, and in the case where the customers closer to them are outside the borders of the zone this would not lead to the best result. Moreover, as specified before, the graph is directed, in order to reflect real roads' fixed travel direction. Dividing it into regions, then, it is not guaranteed that in each area it will be possible to find a path that connects any two internal nodes in it. The assignment of a taxi to a customer in its proximity and in its same station could then potentially result impossible just because a part of the path that connects them goes outside the borders of the region. To solve this second issue and ensure the feasibility of the simulations, another strategy was then implemented, with the aim of reducing the amount of times the shortest path calculation through the Dijkstra method needs to be done to build the cost matrix. For each customer, the Euclidean distance that separates it from every taxi of the network is first evaluated. A certain amount of vehicles is selected among those that result to be the closest to the customer. Only these then go through the next step: the evaluation of the real distance with the Dijkstra algorithm. This method allows to avoid performing the massive time-consuming computation of the shortest path among each customer and a huge amount of taxis that with high probability would not be suitable to be assigned to it, can be avoided. This action made the simulations several times faster, thus definitely more suitable to be applied for a real-time control of the fleet.

3.3.1 Baseline Algorithm

In order to better visualize and evaluate the outcome of this research, it was necessary to compare the performance of the T-EAMoD algorithm with that of a baseline control policy. Unfortunately, it was impossible to find a suitable algorithm for such comparison. In fact the T-EAMoD was designed to be applied to a fleet of autonomous electric vehicles, that can thus potentially run for 24 hours per day, 7 days per week, and for which it is necessary to account for the charging time. A comparison with the performance of a fleet of human-driven fuel vehicles would then not be meaningful, because too many aspects would be different in the two cases. Moreover, as previously stated, the software AMoDeus provides some control policies created by its developers that can be accessed by the users. The problem with such algorithms is that all of them ignore the aspect of battery consumption, thus neglecting the necessity of charging the electric vehicles. In absence of a suitable algorithm to use for the comparison, it was necessary to create it. Such control policy, addressed as "baseline", consists in a reactive assignment of available vehicles to the closest customers, in order to reflect what usually happens nowadays with the classical urban taxi services. The space that separates the vehicles from the customers is evaluated in terms of Euclidean distance, instead of shortest path as instead happens in the T-EAMoD, and also the charging of the taxis is implemented in a reactive way: whenever a vehicle has a battery level lower than a certain threshold, it is put in charge and remains in such state until it has overcome a pre-defined level of SoC. In the following sections, the performance of the T-EAMoD algorithm are compared with those obtained by adopting the baseline control policy to the same city network with the same data set, while in the last Chapter of this thesis the results obtained by running the two simulations in parallel are shown and discussed.

3.4 Real scenario results

The data used for the simulations on AMoDeus reflect the real demand of taxis of the 19th July 2019 in the City of Chicago. These data are described in the population file, where each customer is represented through a request composed by: arrival time, source node and destination node. The population file of the day under test includes 39 590 requests, and its distribution during the day, sampled with the rebalancing period (120 s), is shown in Figure 3.24.

The size of the fleet used for the simulation is equal to the total number of taxis employed in reality during the day to satisfy such demand (990 taxis). The majority of these taxis are fuel vehicles, in opposition to the fleet used in this research that includes only electric cars. This difference is substantial when considering refuelling times. In fact, the time to refuel a petrol car is really short, almost negligible with respect the service time, while the time needed for an electric vehicle to recharge its battery is of the order of a hour. Furthermore, the taxis come from different service providers, whose objective is to maximize their own profit. This behaviour leads to a management of the whole amount of taxis extremely different from the centralized method applied in this research. One of the main breakthrough of this

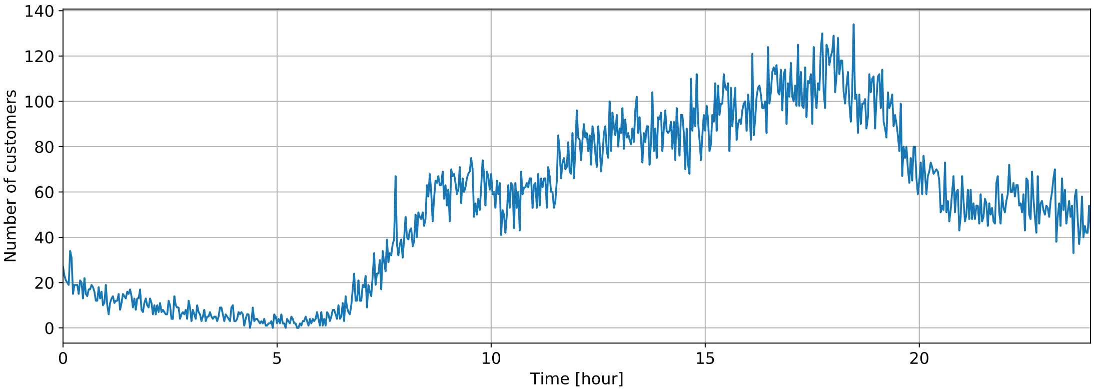


Figure 3.24: *Distribution of the requests during the 19th July 2019 sampled every 120 seconds.*

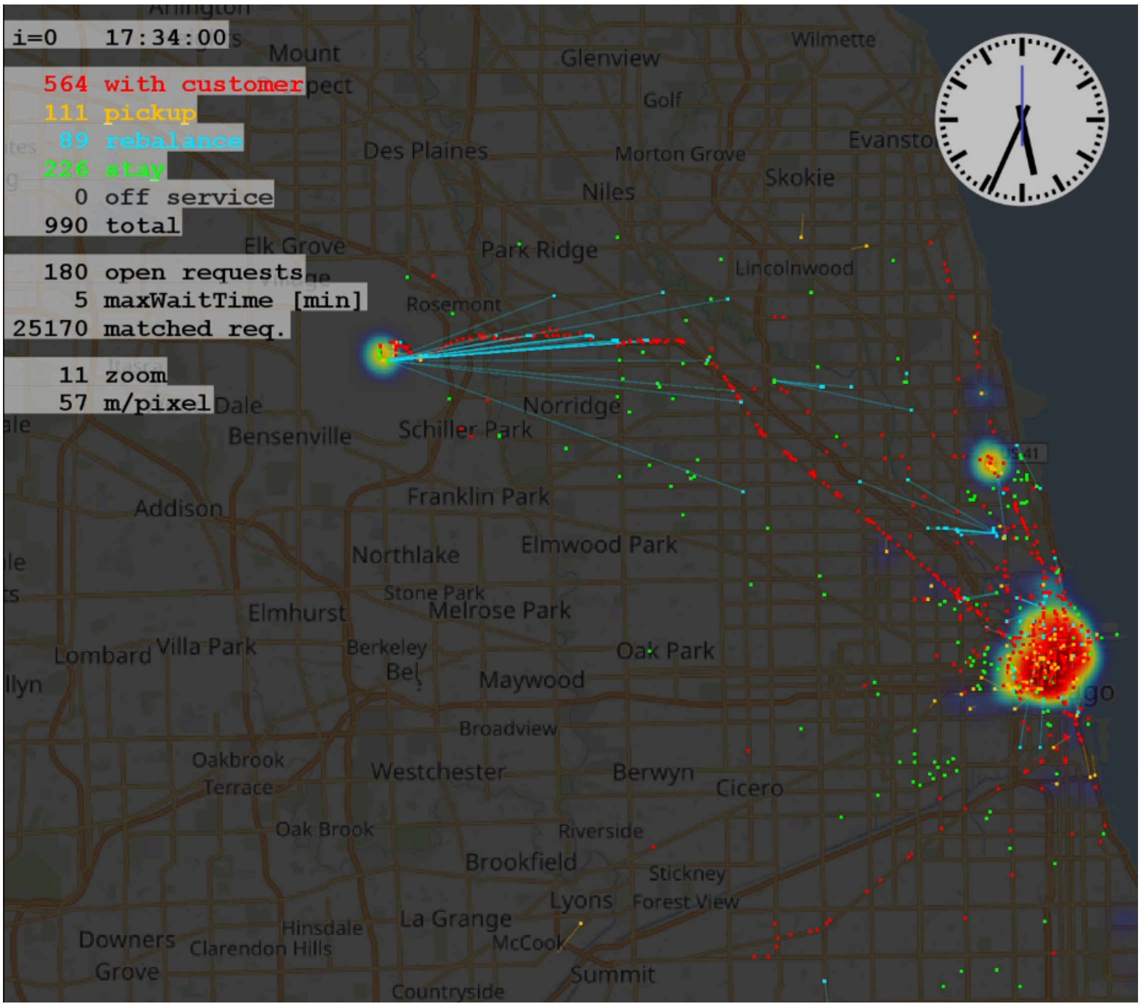
work is the cars' autonomous driving, thus the taxis can be on service 24/7 without any selective working hours. The same could not happen with taxi drivers, in which the number of available taxis would vary as a function of the hours of the day. As just mentioned, this approach has different aspects that are too distant from the current system of taxis, so it would not be meaningful to compare them. For this reason the real performance of the system during the day tested, in terms of waiting time and empty distance, was not taken into account.

Figure 3.25 captures one instant of the simulation with AMoDeus. The green dots correspond to the taxis that are not moving, so they can be either available or in charging state. The red dots instead are the vehicles transporting customers, while the orange ones are those that are going to pick up their customer: the orange lines connect their current position to the position of their passenger. The blue dots correspond to taxis that are busy in a rebalancing trip, and the blue lines connect them with their destinations. The blue regions within the network represent the areas with an elevate amount of open requests and their color tends to red as the number of customers increases.

Comparison of different horizons

As was previously done with the sample network, the simulations on the real network were repeated for different lengths of the horizon H, in order to explore the impact of such parameter on the results and complexity of the simulations. The outcome of this comparison is summarized in Table 3.2.

The first outcome of this analysis that needs to be underlined is related to the baseline algorithm which, as described in Section 3.3.1, tries to reflect what would happen by applying nowadays' taxi fleet management policies on a fleet of autonomous vehicles. Comparing the first two rows of Table 3.2, it is possible to notice a substantial improvement in the system's performance, both in terms of waiting time and empty distance, even in the case without any rebalancing action issued. The explanation that can be given to this behaviour is twofold. First of all, the distance between taxi and customer in the T-EAMoD case is evaluated through a shortest path algorithm



```
i=0    17:34:00
  564 with customer
  111 pickup
   89 rebalance
  226 stay
    0 off service
  990 total

  180 open requests
    5 maxWaitTime [min]
25170 matched req.

   11 zoom
   57 m/pixel
```

Figure 3.25: *Screenshot of an AMoDeus simulation.*

that studies the network itself and thus takes into account all the possible paths that connect their positions. The baseline instead uses the Euclidean method to evaluate the distance between them, which leads to underestimating the real value and thus to a loss of optimality: with that comes a higher distance driven by the empty taxis. The second reason is related to the dispatching step. In the baseline, this phase considers only the distance that divides the taxi from the customer, while in the T-EAMoD algorithm, as described in Section 2.4.2, it takes into account many other relevant aspects as well, such as the waiting time of the customers, the SoC of the taxis and the residual path of the vehicles that are already travelling.

Going back to Table 3.2, and analysing it entirely, it is clear that an extension of H in general brings positive results in terms of customer waiting time reduction: both the mean and the 95% quantile of such parameter decrease when the length of the horizon increases. The value indicated by the 95% quantile indicates the upper bound to the WT of 95% of the customers that use the service along a full day. On the other hand, the total distance travelled by the vehicles grows with H, and the same happens with the empty distance percentage. These behaviors are due to the fact that the more the rebalancer sees ahead in the future, the more rebalancing actions it will issue, and in this way the quality of the service for customers is improved: this cannot happen, at least with the setup used here, without an increase of energy expense of the overall system. A change in the trend can be observed in the values of total and empty distance for horizon lengths of 2 hours and more: the total and empty distance do not increase anymore, they settle around certain values and fluctuate around them, while at the same time bringing benefits in terms of reduction of mean and 95% quantile waiting time.

Horizon	**Mean WT**	**95% quant.**	**Total Dist.**	**Empty Dist.**	**Comp. Time**
Baseline	7 *min* 02 *s*	17 *min* 23 *s*	803832 *km*	208440 *km*	0 *s*
No Reb	6 *min* 14 *s*	14 *min* 49 *s*	788577 *km*	193189 *km*	0 *s*
30 *min*	6 *min* 01 *s*	14 *min* 36 *s*	793281 *km*	197892 *km*	3.65 *s*
1 *h*	5 *min* 51 *s*	14 *min* 06 *s*	799256 *km*	203867 *km*	10.11 *s*
1 *h* 30 *min*	5 *min* 08 *s*	13 *min* 07 *s*	807982 *km*	212599 *km*	21.8 *s*
2 *h*	4 *min* 44 *s*	12 *min* 52 *s*	808791 *km*	213406 *km*	39.02 *s*
2 *h* 30 *min*	4 *min* 41 *s*	12 *min* 34 *s*	810207 *km*	214824 *km*	70.03 *s*

Table 3.2: *Comparison of the results obtained in function of different horizon lengths. The computational time is related only to the rebalancing phase of the algorithm.*

Further clarification about the concept of empty distance are necessary to understand the origin of these values. The distance that a taxi covers without carrying a customer can be divided into two contributions: pickup and rebalancing. The first one depends on the length of the path that separates the taxi from the customer assigned according the dispatching policy, while the latter refers to the trips made in order for the taxi to anticipate the arrival of the customer, thus without being assigned to a specific request. The data of the simulation without rebalacing policy, in which taxis move only when customers appear, better explain the ones showed in Table 3.2. In fact, with the "no-rebalancing" setup, an empty distance of 193 189 *km* is travelled by the vehicles, over a total distance of 788 577 *km*, corresponding to a percentage of about 24.50%. Such setup led to a mean and 95% quantile of the waiting times of respectively 6 min 14 s and 14 min 49 s. In that case the empty distance can be considered as pure pickup distance in fact, without rebalancing, taxis do not move in advance. Thus, from the aforementioned data, it can be deduced how a slight increase in empty distance can lead to sensible improvements in terms of waiting time reduction. Figure 3.26 shows how the ratio between the two components of empty distance varies by increasing the length of the horizon: for a shorter horizon less rebalancing is done, while the percentage of pick up distance among the total empty distance is quite elevate. As H increases, more and more rebalancing is done, and the percentage of pick up travelled distance decreases accordingly.

With this classification of the two contributes that constitute the empty distance,

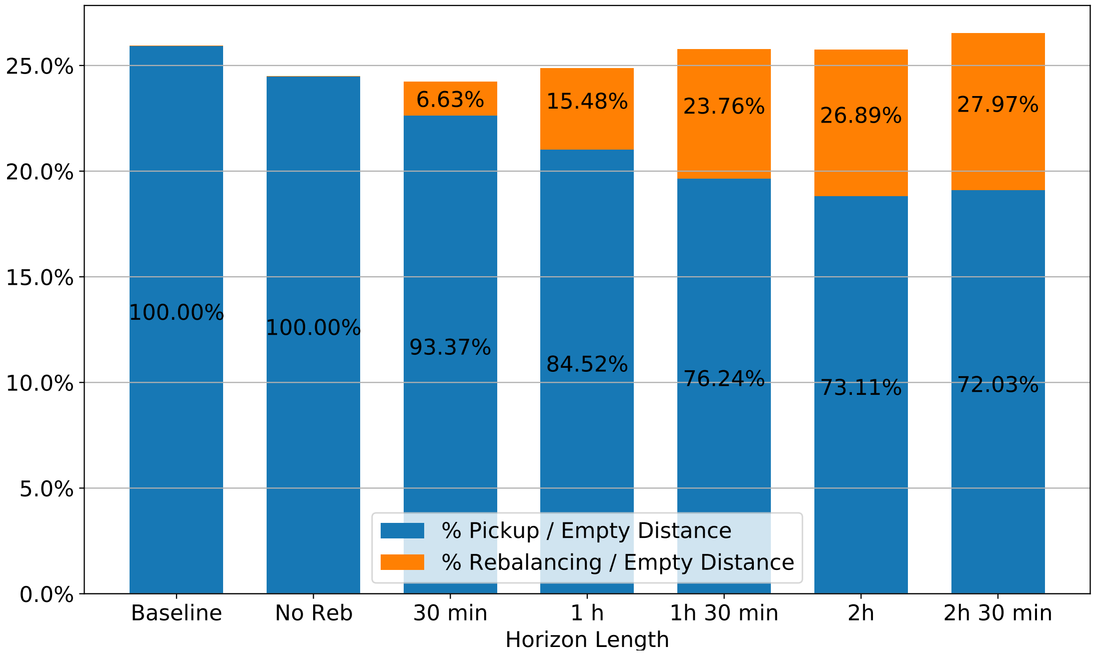


Figure 3.26: *Variation of the ratio of "pickup" and "rebalancing" distance over the whole empty distance with an increase in the horizon length, simulated on the sample network with* $\gamma = 0.7$.

the understanding of the behavior of the curve of empty distance as a function of γ (Figure 3.8, referring to the sample network), parameter which roughly represents the amount of rebalancing done, becomes more straightforward. When the increase of empty distance travelled due to the rebalancing trips is smaller than the decrease of pick up trips, then the curve experiences a deflection. The trend is inverted when instead the rebalancing distance travelled is more than the amount of pick up distance it allows to save. It is also necessary to take into account the increase in computational time for the rebalancing phase caused by an extension of H. Such parameter grows more than linearly with the length of the horizon, and it becomes quite elevate for higher values of H, thus limiting the possibility of increasing the horizon length. In the remaining simulations of this thesis, a horizon length of 2h was adopted, as it seemed to be the right compromise between the necessity of improving the results and that of keeping the computational time under control.

Comparison of the effect of different values of the service parameter γ

...

Comparison of different forecasting methods

Different setups were tested on the real network for what concerns the aspect of demand forecast used in the rebalacing algorithm. All the simulations discussed previously, in fact, were conducted using the known future demand (oracle) as a forecast, because this would allow to better observe the behavior of the algorithm and to tune other parameters without the system being influenced by uncertainties coming from inexact forecasts. In reality however the future demand is not known in advance, so it is interesting to evaluate the effect of a forecasting method on the results of the simulations. Two methods have been tested: AR prediction and frozen demand. The first one, already discussed in Section 2.4, takes as input a batch of data composed by the costumers' source-destination matrices of the n_{batch} previous time steps and, selecting finely the number of lags, it predicts the demand for all the H time steps of the horizon, where usually $n_{batch} > H$. The data forecasted along the whole simulation are showed in Figure 3.27, where they are compared with the oracle demand of each time step. Although the trend of the predicted demand follows well that of the real one, it is clear that the AR sensibly underestimates the data, even after the corrections applied with the robustification step explained in Section 2.4. This might be in part due to the fact that the AR has to predict pairs of values, not singular ones, to account for both the source and destination of the future customers: this makes the data less meaningful and more difficult to predict.

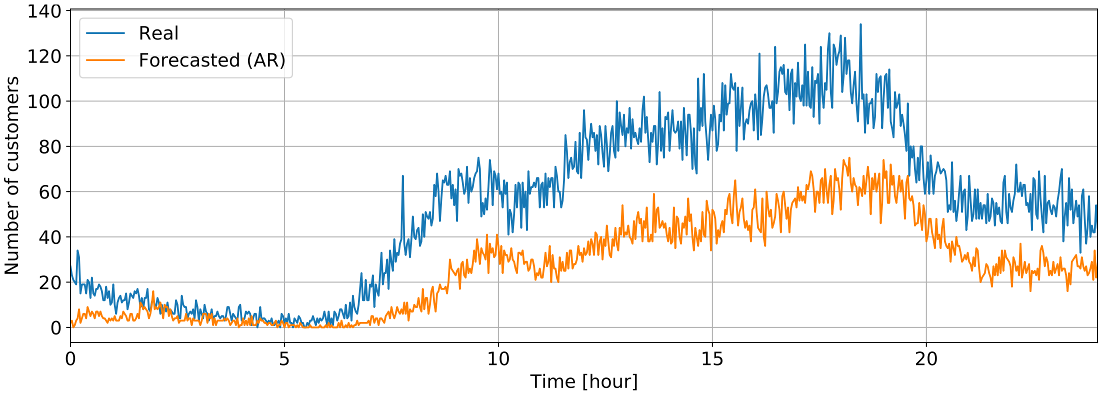


Figure 3.27: *Comparison between real demand and the first time step of the forecast done in each time step of the simulation.*

On the other hand, the frozen demand shows a scenario in which no predictions are made and the customers that appeared between the end of the previous time step and the beginning of the current one replace the forecast for the whole horizon. Table 3.3 compares the main results of these three simulations.

Forecast Method	**Mean WT**	**95% quant.**	**Total Dist.**	**Empty Dist.**
AR	6 min 03 s	14 min 55 s	801890 km	206510 km
Frozen	4 min 56 s	12 min 42 s	903115 km	307718 km
Oracle	4 min 44 s	12 min 52 s	808791 km	213406 km

Table 3.3: *Comparison of the results obtained in function of different forecast methods simulated with* 2 h *horizon.*

The T-EAMoD algorithm is strongly dependent on the quality of the demand forecast. The predictions made with the AR underestimate the real demand: for this reason the waiting time connected to that case are large, while the empty distance travelled by the vehicles remains reasonable. The fixed demand instead works better in terms of waiting time, but leads to more kilometers travelled by empty taxis. In fact, assuming that the demand in the whole horizon will be the same as it is in the present moment, vehicles are often moved to stations where in reality they will not be needed, thus resulting in useless expensive trips. To be able to adopt this algorithm with better results, a more accurate forecasting method would be necessary, which could not be implemented here because it requires studies that go in different directions than the present one.

Reduction of the fleet

Some analysis were performed to explore the possibility of reducing the amount of taxis in the fleet. The comparison was run between the following configurations:

- Full fleet (990 taxis), baseline algorithm
- Full fleet (990 taxis), T-EAMoD algorithm
- Fleet reduced of 5% (940 taxis), T-EAMoD algorithm
- Fleet reduced of 10% (891 taxis), T-EAMoD algorithm

The data in Table 3.4 report, as expected, a decrease in performance as the fleet size decreases. In fact, the simulations with 940 and 891 taxis both have higher mean waiting time and 95% quantile with respect to the simulation with the full fleet.
The most interesting outcome of this analysis is related to the comparison between the results obtained with the T-EAMoD algorithm applied on a reduced fleet and those given by the baseline control policy. A reduction of fleet of 5% in the T-EAMoD leads to an overall customers' waiting time that is still sensibly lower than the one obtained with the baseline. In the case of adoption of the T-EAMoD policy with a fleet of 891 vehicles (fleet size reduced of 10%), the value of the mean customers' waiting time becomes really close to that obtained with the baseline algorithm and a full fleet. As was observed while analyzing the other results of this research, the drawback of this behaviour is always an increase in distance driven by empty taxis. This is reasonable since the demand and the size of the network is kept constant, so the CT rate increases together with the density of the taxis along the network.

Fleet Size	**Mean WT**	**95% quant.**	**Total Dist.**	**Empty Dist.**
990*	7 *min* 02 *s*	17 *min* 23 *s*	803832 *km*	208440 *km*
990	4 *min* 44 *s*	12 *min* 52 *s*	808791 *km*	213406 *km*
940 (-5%)	5 *min* 47 *s*	14 *min* 00 *s*	812324 *km*	216936 *km*
891 (-10%)	6 *min* 59 *s*	15 *min* 14 *s*	819599 *km*	224212 *km*

Table 3.4: *Comparison of the results obtained in function of different fleet sizes. Simulated with 2 h horizon. *Simulation with baseline algorithm*

Having a smaller amount of vehicles would result beneficial in terms of reduction of the costs connected to the production, maintenance and management of the fleet.

A service provider might then accept a slight increase in customers' waiting time and energy expense connected to the vehicles' trips, in order to take advantage of the benefits that would come from the fleet reduction. With a detailed study of the cost savings and of the increase in efficiency that come from having a reduced fleet, it would be possible to find the optimal fleet size, keeping in mind that the optimization of the dispatching, rebalancing and charging phases would still allow to offer a good service quality, by reducing customers' waiting times compared to the baseline that reflects the reality of taxi fleet management nowadays, even with a lower amount of vehicles.

4

Conclusion

4.1 Final results

The detailed analysis conducted in Chapter 3 shows the effect of all the parameters that influence the system, their physical meaning and how they can be tuned in order to obtain the desired objective. As an outcome of the research, a final simulation was done adopting the set of parameters that, according to the analysis carried out previously, seemed to bring to the most interesting and reasonable results. In particular, the values adopted are the following:

- length of the time horizon: $H = 2$ hours,
- service parameter: $\gamma = 0.7$
- number of vehicles: $N_T = 990$

The optimization problem solved in the rebalancing phase is shown in Equation 2.12, which includes the cost function where the weights $w_{z_i}^t$ of the decision variables related to the charging taxis z_i^t are all equal to zero, the value of $\widetilde{y_i}^t$ is set to zero and the imbalance term is squared. The forecast is once again replaced with the real future demand (oracle) because, as was assessed by the analysis on the Autoregressive model results, other forecasting methods need to be investigate to find the most suitable one for the present application. The performance of the T-EAMoD algorithm was compared with that of the baseline described in Section 3.3.1, in order to better evaluate the results and the improvement that the T-EAMoD control policy brings in the field of AMoD control.
The full simulations are visible here [37], and a frame extracted from them is shown in Figure 4.1. The instant of the simulations visible in the figure corresponds to a moment with a quite elevate customer demand, as testified by the coloured areas on the network. Those represent the open requests in real time: they are blue when the amount of customers in the zone is relatively low, and tend to become red as the amount of people that are waiting increases. During the simulation, in the case of the optimized algorithm those zones are on average smaller and lighter than what happens with the baseline, meaning that the rebalancing anticipates quite well the demand, and thus there is a lower accumulation of customers even during the peak hours.

Some significant results are also shown in Table 4.1. The average customers' waiting time obtained with the T-EAMoD algorithm is of 4 minutes and 44 seconds, resulting to be 33% lower than the baseline's, while the 95% quantile is reduced of 26%. Such improvement in performance costs to the system an increase of only

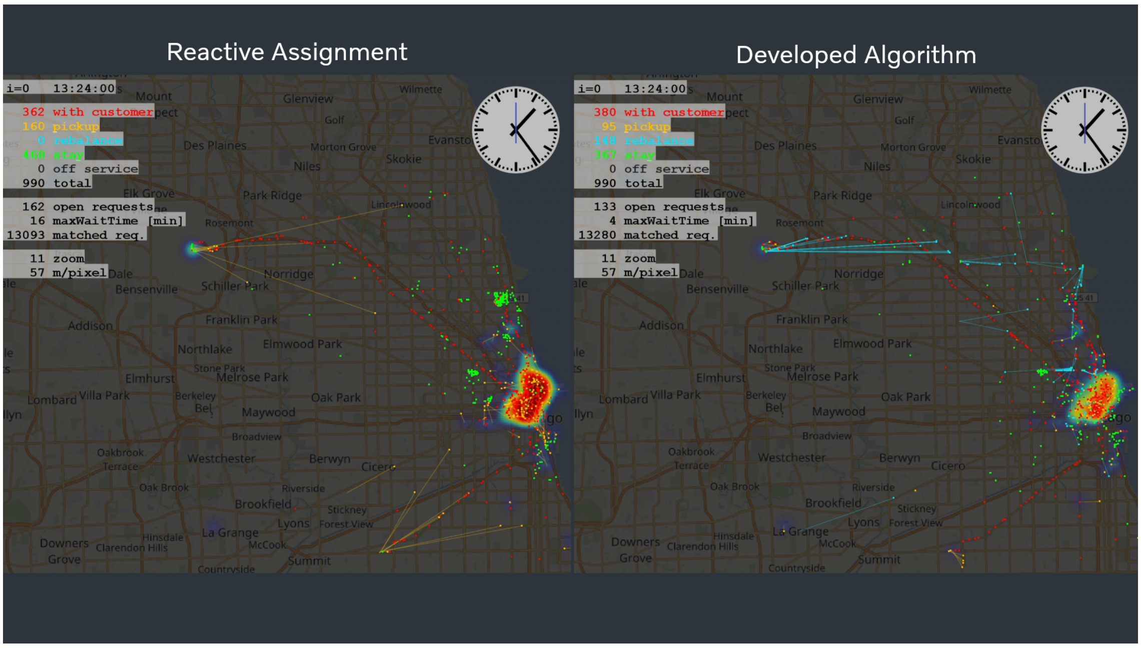


Figure 4.1: *Frame of the comparison between the simulation with the T-EAMoD control policy and the one with a baseline reactive algorithm.*

Control Policy	Mean WT	95% quant.	Total Dist.	Empty Dist.
Baseline	7 *min* 02 *s*	17 *min* 23 *s*	803832 *km*	208440 *km*
T-EAMoD	4 *min* 44 *s*	12 *min* 52 *s*	808791 *km*	213406 *km*

Table 4.1: *Results of the comparison between the simulation with the control policy developed in the present research and the one with a baseline reactive algorithm.*

2.4% of distance travelled by empty vehicles, thus suggesting that the T-EAMoD is effective in managing the whole system in a more optimized way. The only aspect on which the T-EAMoD performs worse than the baseline algorithm is obviously the computational time: the time needed by the T-EAMoD to evaluate the data and produce the optimal control actions at every time step is orders of magnitudes higher than that required by the baseline for the same actions, so a more powerful data processing machine is necessary in general to run the T-EAMoD policy.

4.2 Discussion

The algorithm illustrated in this research can adapt to various scenarios and be oriented towards different objectives, according to what is the main goal that the company providing the AMoD service wants to pursue and what are the resources it can rely on. As discussed in Section 3.2, the user can easily decide to prioritize either the empty distance travelled by the vehicles or the customers' waiting time, by accordingly tuning the service parameter γ. The value of $\gamma = 0.7$ seemed a reasonable trade-off between the two objectives, and was adopted here to present other kinds of results, but it can be simply reduced for a system whose main goal is a cut in energy expense, or increased in case customers' satisfaction needs to be boosted, at the expense of energy saving.

Moreover, a semi-realistic model of charging was adopted, with the aid of some simplifying assumptions that neglect the nonlinear behavior of the electric vehicles' SoC, which anyways do not cause large errors until the battery level remains in the range 20 - 80%. The model can be improved to bring more accurate results, but the main effects of the introduction of the charging aspect in the algorithm are clearly visible and provide an interesting outcome. The method chosen to include it in the model was to incorporate it into the optimization problem of the rebalancing, thus to address it in a predictive framework. This was proven to provide considerably better results compared to the option of dealing with charging just in a reactive way, as it allowed to handle the charging of vehicles in a smart way, to avoid any interference with the demand peaks. The fact that the charging decision variable z was also included in the cost function expression leaves space to the introduction of aspects that concern energy price and availability fluctuations over time, that here were not considered.

Several studies about the optimal horizon length were also performed. Their results show that the choice of H must remain between some predefined limits, the lower bound being given by the minimum number of time steps needed in order for the predictions to be meaningful, while the upper one is caused by the increase in computational burden given by an extension of H, which limits the method in terms of feasibility. An extension of the horizon in general is expected to bring an improvement in the results only until a certain value of H, and after that the outcome of the application of the algorithm is not influenced by an increase in horizon length. The optimal solution would be to adopt such value, which differs according to the system setup, after which the dependence of the results on H is not observable anymore. In case of availability of more powerful resources than the ones used for this thesis, such value of H can be better investigated in a real scenario and adopted, thus improving the quality of the fleet management.

4.3 Future studies

Research in the field of AMoD control started in the recent years, and some of its key issues have not been addressed yet. This thesis in particular, besides suggesting solutions to some interesting questions on the topic, raises new issues and indicates some possible new directions that researches on AMoD could follow in the future. What would be interesting to implement, for example, is a multiple simultaneous Hungarian algorithm for the dispatching phase, that would allow to assign a taxi to more than one customer in the same time step, arranging them in a temporal queue. This could be useful in case the destination of a passenger corresponds to the source of another: energy could be saved by avoiding sending a vehicle to a station where another one is already going, even if customers' waiting time might be slightly penalized. For what concerns the charging aspect, a more accurate model could be built to account for energy consumption, when the exact model of the vehicle used is known as well as the road conditions and congestion. The traffic in general is an aspect that needs to be taken into account in future studies, because it also impacts the travel time and thus the assignments that can be made. Moreover, it would be interesting to define the exact position and capacity of the charging stations, which would add some more constraints in the algorithm. In that case, the distance that the vehicles would need to cover to reach the charging stations would also have to be included in the count of the overall empty distance. Talking about demand forecast, different interesting models could be investigated, in particular neural networks could be involved in it, as they could be properly trained on data taken from the past and provide more accurate forecasts compared to the Autoregressive models. For what concerns the implementation on the software AMoDeus, different ways of partitioning the network could be investigated. For example, it would be possible to adopt the technique of dynamically partitioning the network to follow the behavior of the demand and its distribution along the network, in order to make the algorithm more efficient. An interesting expansion of the study involves the aspect of ridesharing, in other words the transportation of multiple customers, each with his/her own source and destination, by the same vehicle at the same time: this would definitely optimize the efficiency of the AMoD system by further reducing the distance travelled.

4.4 Conclusion

In this study, an algorithm for the control of Autonomous Mobility on Demand oriented not only in the direction of an increase of the service quality but also towards energy efficiency was presented. Different aspects needed to be taken into account when planning the dispatching and rebalancing phases of the fleet management, and besides finding the optimal solution to service all the customers in the shortest time possible, it was necessary to consider also the aspect of vehicles charging and distance travelled by empty taxis. Electric vehicles in fact have the capability of reducing the environmental impact of the transport sector, but in order for this to happen their utilization must be optimized. The time to charge an electric vehicle, in fact, needs to be taken into account: it is not as short as the time needed to refuel petrol cars which, instead, could be neglected. Consequently, adopting electric vehicles without an appropriate smart policy to account for their charging would mean that, for a fixed amount of average demand, a much higher number of cars would be needed, with repercussions on the environment as well as on the congestion of the streets. The method presented in this thesis was meant to optimize the system taking into account as many aspects as possible, including: the quest for the best taxi-customer matching, the need of balancing the resources along the networks, the opportunity cost connected to the charging of vehicles, the customers' whole trips, including not only their starting positions but also the destinations. The outcome of the study shows useful results for the development of an optimal AMoD control policy, but at the same time leaves much space for orienting the policy towards the most desired objective by appropriately tuning the system parameters, whose impact on the results was widely discussed.

Bibliography

[1] Mobility and Transport - European Commission. https://ec.europa.eu/transport/themes/its/road_it.

[2] NHTSA's National Center for Statistics and Analysis. 2016 fatal motor vehicle crashes: Overview. *Traffic Safety Facts, US Department of Transportation*, 10 2017.

[3] The Evolution of Automated Safety Technologies | NHTSA. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[4] Marco Pavone. *Autonomous Mobility-on-Demand Systems for Future Urban Mobility*, pages 399–416. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[5] Uber. www.uber.com.

[6] Lyft. https://www.lyft.com/.

[7] Bolt. www.bolt.eu.

[8] Aimo Solutions. https://aimosolution.com/?lang=en.

[9] M. www.m.co.

[10] CAR2GO Group GmbH. https://www.share-now.com/.

[11] Share'ngo Slovenija. https://site.sharengo.si/en/.

[12] Rick Zhang and Marco Pavone. Control of robotic mobility-on-demand systems: A queueing-theoretical perspective. *The International Journal of Robotics Research*, 2016.

[13] Todd Litman. Autonomous vehicle implementation predictions: Implications for transport planning. 2020.

[14] ZOOX. https://zoox.com/.

[15] EasyMile | Autonomous vehicle technology and solutions. https://easymile.com/.

[16] Sebastian Hörl, Claudio Ruch, Felix Becker, Emilio Frazzoli, and Kay Axhausen. Fleet operational policies for automated mobility: A simulation assessment for zurich. *Transportation Research Part C Emerging Technologies*, 102:20–31, 05 2019.

[17] Joschka Bischoff and Michal Maciejewski. Simulation of city-wide replacement of private cars with autonomous taxis in berlin. *Procedia Computer Science*, 83:237–244, 12 2016.

[18] Lukas Sieber, Claudio Ruch, Sebastian Hörl, Emilio Frazzoli, and Kay Axhausen. Improved public transportation in rural areas with self-driving cars: A study on the operation of swiss train lines. *Transportation Research Part A Policy and Practice*, 134:35–51, 02 2020.

[19] R. Zhang, F. Rossi, and M. Pavone. Model predictive control of autonomous mobility-on-demand systems. 2016.

[20] Ramon Iglesias, Federico Rossi, Kevin Wang, David Hallac, Jure Leskovec, and Marco Pavone. Data-driven model predictive control of autonomous mobility-on-demand systems. 2017.

[21] M. Tsao, R. Iglesias, and M. Pavone. Stochastic model predictive control for autonomous mobility on demand.

[22] Matthew Tsao, Dejan Milojevic, Claudio Ruch, Mauro Salazar, Emilio Frazzoli, and Marco Pavone. Model predictive control of ride-sharing autonomous mobility-on-demand systems. pages 6665–6671, 05 2019.

[23] Riccardo Iacobucci, Benjamin McLellan, and Tetsuo Tezuka. Optimization of shared autonomous electric vehicles operations with charge scheduling and vehicle-to-grid. *Transportation Research Part C: Emerging Technologies*, 100:34–52, 2019.

[24] Fei Miao, Shuo Han, Shan Lin, John A. Stankovic, Hua Huang, Desheng Zhang, Sirajum Munir, Tian He, and George J. Pappas. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *CoRR*, abs/1603.04418, 2016.

[25] Hong Cui, Jingjing Zhang, Chunfeng Cui, and Qinyu Chen. Solving large-scale assignment problems by kuhn-munkres algorithm. 01 2016.

[26] J.M. Maciejowski. *Predictive Control: With Constraints*. Prentice Hall, 2002.

[27] Autoregression Models for Time Series Forecasting With Python | Machine Learing Mastery. https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python.

[28] Autoregressive models | Forecasting: Principles and Practice. https://otexts.com/fpp2/AR.html.

[29] Autoregressive model | Wikipedia. https://en.wikipedia.org/wiki/Autoregressive_model.

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

[31] Dell Inspiron 7559 Specifications. https://downloads.dell.com/manuals/all-products/esuprt_laptop/esuprt_inspiron_laptop/inspiron-15-7559-laptop_reference%20guide_en-us.pdf.

[32] AMoDeus. https://www.amodeus.science/.

[33] MATSim. https://matsim.org/.

[34] Bidirectional Dijkstra shortest path | NetworkX. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.shortest_paths.weighted.bidirectional_dijkstra.html.

[35] Volvo XC40 | Volvo Cars. https://www.volvocars.com/intl/v/cars/xc40-electric.

[36] Dijkstra's algorithm | Wikipedia. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Running_time.

[37] Simulations | YouTube. https://youtu.be/IUHJf4nt0Dc.