Doctoral Dissertation
Doctoral Program in Electronic and Telecommunication Engineering (33.th cycle)

Electric Revolution and Free Floating Car Sharing

A Data Driven Methodology for System Design

Michele Cocca
* * * * * *

Supervisor
Prof. Marco Mellia, Supervisor

Politecnico di Torino
January 20, 2021

. . . .

Michele Cocca
Turin, January 20, 2021

Summary

Nowadays, the increase in traffic congestions, land consumption, and pollution emission due to private car ownership makes the rise of shared mobility possible. One of the most spread implementations of shared mobility is Free Floating Car Sharing (FFCS). It is a car rental model where the users can pick and release the car everywhere within an operative area. The customers can reserve (and return) the vehicle using a web-based application. With just a simple tap, the users can unlock and lock the smart vehicle. Usually, the provider bills the users only for the time spend driving, with time-minute based fares. All the other costs, like petrol, insurance, and maintenance, are in charge of the provider.

This service's flexibility fills the urban mobility gap between public transport's relative cheapness and the comfort and capillarity of private car ownership. Indeed, FFCS allows people to travel and commute faster than the standard public bus but avoiding all the fixed and variable costs related to private car ownership.

Given the recent electric cars market increase and all the benefits those vehicles carry, replacing FFCS fleet with electric-powered cars may still improve urban centers' quality of life. The setup and management of an electric FFCS require ingenuity to minimize the users' discomfort due to car plugging procedures.

In my thesis, I present a methodology to address, in different cases of studies, all the challenges related to the conversion of combustion engine cars to electric vehicles in FFCS. In particular, my research's main driver is to propose a methodology to build a profitable and technically sustainable system setup, able to guarantee a flexible and appealing mobility service to an increasing customer audience.

In the first part of my thesis, I describe the software I developed to scrape from the web real combustion engine FFCS, from two providers: car2go and Enjoy. The car2go data collection lasted from December 2016 to January 2018, collecting more than 27 million users' bookings spread in 23 cities. The Enjoy data collection phase started in May 2017 and lasted until June 2019, recording about 6 million bookings in 6 cities.

Then, I characterize both datasets in Turin, one of the cities in which both FFCS providers work. I detect the outliers, filter them out from the dataset, and extract geo-temporal users' travel patterns.

After that, I compare the car2go customer's pattern with the one-way and two-way car-sharing system. The results show how users prefer more flexible services like FFCS or one-way car sharing.

Once the data are consolidated, I develop: A methodology to place a charging station in a city by looking at users' patterns. System policies to manage the fleet when the vehicle state of charge may not guarantee a trip. Via an event-based trace-driven simulator able to replicate the recorded trips in an electrified scenario evaluating each configuration's feasibility.

Via accurate simulation in Berlin, Milan, Turin, and Vancouver, I study different electric FFCS setup. By placing the charging station in the most frequented areas, by offering an incentive to the users to plug the car when the battery state of charge is below a safety threshold, and balancing the spread of poles, it is possible to obtain a sustainable system covering with charging station only the 8-10 % of zones.

To reduce the number of charging stations to have a sustainable electric FFCS, I compare several optimization algorithms. The results show how a Genetic Algorithm can find a better solution to shrink the minimum amount of resources to sustain the same mobility demand.

After that, I move my attention to the users' rentals' demand predictability. The main goal is to understand how different open-data sources could impact the recorded FFCS users' rental. Initially, I compare several time-series forecasts to predict the users' demand in the short and medium-term. Random Forest regression produces better accuracy and results in terms of interpretability. Then I correlate the socio-economics features characterizing each city neighborhood to FFCS demand, and again, the Random Forest regression outperforms other algorithms.

Finally, I question the system scalability figuring out several scenarios having increasing demand. I use a model to synthesize users' demand by looking only at the geospatial users' rentals. By varying the electric FFCS setup and simulating the new scenario, I point out how a linear increase in the demand intensity requires a fleet sublinear increase. Finally, I project those considerations in euros, proofing how electric FFCS has room for economic growth.

Acknowledgements

If someone asked me five years ago where I would be today, I would never imagine myself here, writing my Ph.D. dissertation. Therefore, in this third thesis, it is necessary to thank all the people that made it possible to reach this goal.

Initially, I would like to thank Marco, my advisor, for continuously mentoring me with your insights and your trust in me and my skills. A special is due to the reviewer, professor Anna Maria Vegni and Shlomo Bekhor that with their suggestions, improved my work a lot. Then, a special thanks to Danilo, Luca, Martino, and Idilio, the *Esperti* with which I shared the path along this journey. A special mention to the D8RATTI: we did not work together, but coffee breaks with you were fundamental during this time. So a huge thanks to Andrea B., Andrea M., Giuseppe, Marco, and Greta. Other special thanks to Flavia, Alessandro C., and Andrian, for the unmissable brainstorming and support (since Projects X's time) even when we were separated by the Ocean or by irreconcilable points of view about life. Other thanks to Dena and Alessandro D. and their will to learn Piedmountes and their suggestion and support that helped me during those years. Finally, a special thanks to Franca, the colleague who everyone needs but, for sure, I do not deserve.

Looking around, I have to thank all the other people coming from outside the working environment. First of all, I have to mention all the rugbymen and rugbywomen I encountered during my abroad experiences. So, thanks to the Belo Horizonte Rugby Clube, who behaves as a family even when my family was 9000 kilometers far away. Thanks to Rugby Club Luxembourg for the short but intense season. Last but not least, all the people from Settimo's A&P group and Rivoli Rugby A.S.D., my past and current club, for having endured with patience all the times I skipped work outs to work on my Ph.D. Thanks To Monica B., for our physiotherapy and psychological sessions.

Outside sport, I have to say a huge ”thank you” for the laugh and the support to Marco R., Luca D. V., Tommaso C. and Riccardo P. during our discussion about the history and news. Thanks as well to the NMC & affiliates for the continuous support and the trust in the Cocca index. Finally another thank you to 011 GSoM, we grow up together and we still argue like 20 years ago, but your support motivated me to reach this goal.

Thank to all of you.

A special thanks to Deborah for always believing in me and what I was doing, even when I didn’t.

In the end, the biggest and most significant thanks to Mamma e Papà.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, the 55% world population is mainly concentrated in urban centers, and the authors of [1] forecasted, in 2018, an increase of this estimate until 68% before 2050. In this scenario, one of the problems that afflict big conurbations is mobility. Indeed, population growth as well as the mobility demand. In Europe, this effect is particularly emphasized, and the past flourishing automotive industry satisfied the demand letting the users buy their own car [2].

The increase of private vehicles derived from the growing mobility demand leads to a higher probability of traffic congestion, especially in rush hours. A lot of past and recent studies demonstrated how traffic jams have a negative effect on public health. For example, [3] showed how the micro dust concentration is strongly related to traffic, measuring air pollutant before and during a truck strike. Moreover, [4] computed the monetary impact due to PM2.5-related emission attributable to congestion. The results are shocking: public health may cost up to $17 billion in 2030 in the United States. Another aspect that contributes to a decrease in the quality of life in a congested city is noise pollution. In particular, [5] and [6] demonstrate how intense noise emissions can cause permanent damages to people living close to congested areas. Finally, the problem of land use related to car presence is another critical point that must be taken into account.

Several different regulatory boards addressed the traffic congestion problem in the past years, increasing and optimizing the road network. According to several works like [7, 8, 9], adding more resources attracts more car, actually feeding the traffic congestion and thus increase related the damages.

The recent rise of shared economies made possible a groundbreaking revolution in mobility too. In particular, this paradigm allows a customer to access goods and services through a peer-to-peer instance. More in detail, the shared economy is characterized by the shared creation, production, distribution, trade, and consumption of goods and services by different people and organizations. In other words, an instance of a good or service can be use used by different customers.

Thanks to this particular business view, the so-called *car sharing* born. The first implantation was a shared car purchase among people who could not afford it. During the economic growth after WWII, the shared cars increasingly appeal, and a lot of business started across all Europe, like [10], and [11] tells.

With Internet advent, car-sharing still improved the users' experience. In particular, the providers' IT infrastructure made it possible to reserve and release the car using a web-based applications run on smartphones. In contraposition with the classical car rental model, with per-day fares, the new technologies made possible to bill the users' with new fares based mainly on time spent driving.

Finally, it is possible to define what is intended as **car sharing** today: *it is a car rental model, where the users pay only for the time spent driving, all the other costs, like petrol, insurance and maintenance are in charge to the provider. The reservation and return producers are possible without the customer's physical presence in the provider front-end crew office.*

During the years, several implementations of car-sharing were born. A definition of all car-sharing typologies is needed to highlight the differences among them. In the first approach, It is important to introduce the concept of **operative area** (common to all the car-sharing typologies): the portion of city area over which the users can release the car. Usually, it is as big as the municipality where the users commute.

The main typologies of cars-haring are:

- **Two-way car sharing**: The users can reserve a car in one of the several ad-hoc spots spread around the operative area, but they are forced to return it in the same spot. It allows the providers to limit the fleet management operations to meet the demand but heavily limits the service flexibility reducing the freedom degree of the users.

- **One-way car sharing**: The user can reserve a car in one of the company-owned parking stations, but contrary to the previous one, she/he can return the car in a different parking station. This approach increases the service flexibility, but it requires more fleet load balance by the provider.

- **Free-floating car-sharing (FFCS)**: The users can pick and release the car *everywhere* within an operative area. It means that shared cars can be parked in common parking like private cars. In this case, the provider does not build and infrastructure, and consequently, the fleet management is fundamental to maximize the match between users' demand and current car position.

Since the primordial implementation of car-sharing intended as a business model become operative, the debate on this service's environmental sustainability was very spirited. In particular first studies at the beginning of the millennium, like [12] analyzed the benefits costs of car sharing set up in the UK. It points out how the environmental benefits due to the decrease of private cars circulation like CO_2 emission and land use were achieved, with a fraction of the cost needed to design new road scheme.

Later, [13] investigates the possible public transport abandon by customers due to the presence of this new alternative in urban mobility. Counter-intuitively the author proved that car-sharing fills the users' mobility demand between public transport and private car driving, reaching thus the condition in which it is possible to give users the benefits of both transport modes. Thus, by including car-sharing in the ecosystem of urban mobility offers, the users will be more prone to avoid to travel on their own car and finally incrementing the audience of public transport users.

The milestone event in shared motility was the launch of car2go in 2008 in Germany. This was one of the first completely automatized Free Floating Car Sharing providers. As previously mentioned, Free Floating Car Sharing allows users' to pick and return the car without any constraint. This flexibility attracted more customers: car2go started in few cities in Germany, extended its services in 2017 in more than 26 cities in Europe and North America. Recent studies, like [14], still confirm that the presence of free-floating car-sharing, like car2go, leads to a decrease of car ownership and the consequent increase of life quality in urban centers.

However, it is possible to think a step forward in environmental sustainability. Indeed, the majority of the cars belonging to car2go have an internal combustion engine. The electric revolution we are living nowadays may lead to several improvements to this service too. The benefits due to mobility electrification are well known. In particular, some works like [15] and [16] confirms that electric vehicles can still reduce the concentration of the pollutant in big cities.

The electrification of shared mobility is at the basis of this work, whose main research question is:

It is possible to design an electric car-sharing system keeping the free-floating paradigm, namely without forcing the users to park the car in a proper charging station at the end of each ride?

In electric mobility and particularly in an electric shared mobility scenario, the burden of the recharging operations plays a fundamental role in terms of service attractiveness. For example, a Telsa Model S can require between 13 and 17 hours for a complete recharge if plugged into a domestic power supply [1], which is an unacceptable time if compared to the few minutes that petrol fills up requires.

It follows that free-floating car-sharing providers have to carefully design their electric infrastructure. In particular, the amount and the electric charging station placement play a key role in this scenario. If the system is correctly sized, it will make it possible to have a more customer-centric service able to provide to the customers a *flexible* shared mobility experience.

In order to tackle this problem, in this thesis, I designed a methodology able to replicate the shared mobility demand in a scenario with electric cars defining the Key Performances Indicators (KPIs) that evaluates the sustainability and profitability of an electric

[1] https://www.tesla.com/it_IT/support/home-charging-installation

Free Floating Car Sharing System. Thus, the complete pipeline will provide useful insight into the charging station placement, the perceived users' discomfort related to the plugging operation and customers management policy, and the whole system's profitability.

The structure of this thesis reflected the pipeline flow to design an electric shared system. More in details, in chapter 2 I described the software architecture through which I collect about 35 million users' rides from two Free Floating Car Sharing Provider. Then, in chapter 3 I characterized the two services and the users' habits using a Turin as a case of study city. After that, in chapter 4 I compared the different implementation of car-sharing to understand different users' patterns. In chapter 5 I describe the core of the whole thesis. It is an ad-hoc trace-driven electric free-floating simulator. It takes into input the lists of users' ride and replicates the same demand in a scenario with electric vehicles. Here I define as well all the KPIs that measure the system efficiency. Chapter 6 discusses the charging station placement algorithm driven by the collected data comparing the performances of those placements with the simulation outputs. Chapter 7 compares different optimizations algorithms the charging station placement pointed out by the previous chapter. Then, chapter 8 studies the predictability car-sharing users' demand, discussing which temporal and socio-economic features may influence the Free Floating Car Sharing Demand. Subsequentally 9 compares the system performances tuning some key inputs like the demand intensity, the fleet size, and the infrastructure projects the results on the profit plan from a business point of view. Finally 10 concludes the thesis.

Chapter 2

Data Acquisition Pipeline

This chapter refers mostly to my paper ” *UMAP: Urban mobility analysis platform to harvest car sharing data* [17], presented at the 2017 IEEE SmartWord conference. My contribution is mainly related to the design and the implementation FFCS providers' web scrapers.

2.1 Introduction

This chapter describes how the Urban Mobility Analysis Platform (namely, *UMAP*) collects, processes, augments, and stores real FFCS data and makes the them available for further analyses. In particular, I build two crawlers to collect data from the *car2go* and *Enjoy* platforms[1]. Every minute, the crawler checks which cars are currently available. Every time a given car "disappears", it records the booking start time. The same booking ends when the crawler sees the car available back on the system. Some booking is actual "rental" in case the car moved from the prior parking position to another. Ingenuity must be used, e.g., to filter GPS fix issues (which may erroneously let a car "move"), or to handle possible data collection issues (e.g., the website going down, or some cars undergoing maintenance), or platform design (e.g., synchronous or asynchronous updates).

In total, *UMAP* collected about 27 million trips in 23 cities for car2go, working from 2017 to 2018. Instead, considering Enjoy, I get about 6.6 million rides from 6 cities from May 2017 to June 2019. The source code of *UMAP* for research purposes.[2]

The remainder of this chapter is structured as follows: section 2.2 describes in detail the raw data structure and the data flow from the providers' API to the middle stage. Section 2.3 describes how the software implements the raw data elaboration to trip

[1] `www.car2go.com`, `enjoy.eni.com`

[2] `github.com/MobilityPolito/`

records and their storage into the data lake. Section 2.4 introduces the final stage of the data collection pipeline: a model able to perform several statistics on the collected data. Finally, section 2.5 concludes the chapter.

2.2 Data Acquisition

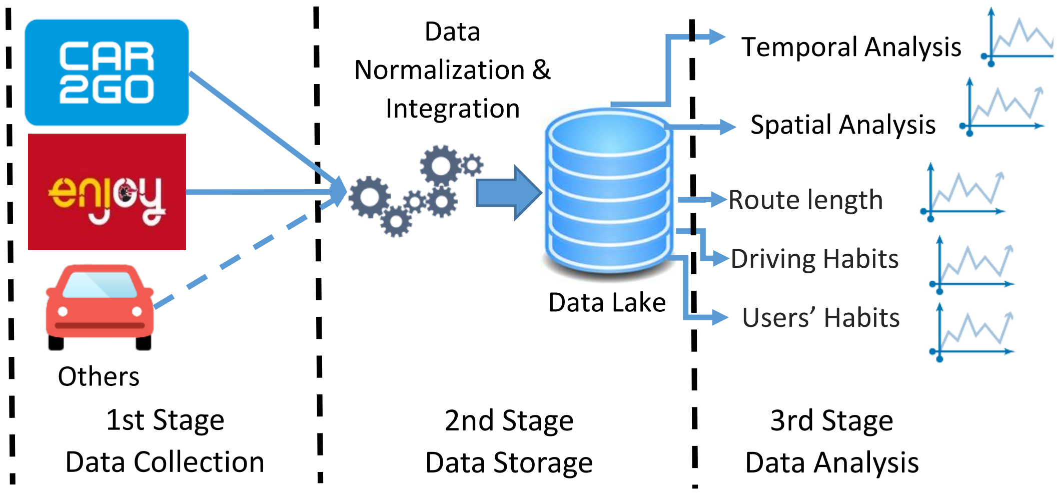


Figure 2.1: *UMAP* overview

In this section, I provide a description of *UMAP* structure. Figure 2.1 depicts the architecture of *UMAP*, composed of a first module for the data acquisition, a second module for data normalization and integration, and then a third module for the data analysis.

The first module consists of the data acquisition from the car-sharing platforms of interest. These typically expose information about cars' location when available for rental through a web-service approach.

For this module, I design two crawlers, one for the car2go and one for the Enjoy car-sharing platforms. They retrieve, at each time instant, which cars are available in a given city.

While car2go offers public APIs [18], Enjoy does not provide to users such a service. For this reason, I reverse engineer the Enjoy web portal. By leveraging the Chrome Developer Tools, I investigate the information exchanged with the Enjoy web portal while asking for the list of available cars. Through this analysis, the software obtains both the URL used to request the list of available cars and how to fetch the data for a specific city. Both systems return the currently available cars using a JSON file.

Each time the system downloads a JSON, a *snapshot* describing which cars are parked and ready for rental. Basically, the *snapshot* is a list containing all cars and their attributes.

In a nutshell, a car is described by the car-sharing web-service as an object annotated by several information, like the plate, vehicle identification number (VIN), location, fuel level, model, etc. All the data represented in this object is useful for the customers, e.g., to choose which car to rent. This object is only present if the car is available, i.e., it is parked and free for a rental. Its state changes over time. In particular, a car disappears when a customer reserves and rents it, and then it reappears when the customer ends the rental (likely in a different location).

At each time t, the software gets the JSON snapshot S listing the available cars. The sampling period has been set to one minute to balance the aggressiveness of the crawler and a reasonable time resolution. S describes each available car with several fields, some of them being in common between the considered companies, but in general with a different format. For this study, I collect each car's unique identifier and the current geolocation indication. These are obtained from the *VIN* or *plate* field, and the *coordinates* field which describes the *longitude* and the *latitude* of the in-car GPS used to localize it when parked.[3] In addition to these fields, the car-sharing JSON description may provide other information, e.g., the *street address* corresponding to the coordinates, the *fuel* level, the *car interior status* the *engine type*, etc. Since each platform uses its own data and format, I design a data integration step to have common names for fields containing the same information, if present.

2.3 Data Normalization and Integration

In this second module, I illustrate how *UMAP* processes and consolidate each snapshot to obtain *parking* and *bookings* periods for each car. A *parking* is the time where the car is available for a user ride. On the other hand, a *bookings* is the time between two parking where the car is not tracked by the system. The intuition is to track the availability of each car on the car-sharing platform, and rebuild the historic parking and booking periods over time: when a customer books a car, the latter "disappears" from the system. *UMAP* records this event, with the initial time and position of a new booking. When the customer ends the booking, the car "reappears" in the system. *UMAP* records this event, deriving the final time and position of the booking. For the same car, a new parking period starts.

Harvested data is unstructured and may grow large. Thus I leverage *MongoDB*, a NoSQL document-based database. A MongoDB database includes set collections, i.e., groups of documents. Each document is a set of key-value pairs, organized in a JSON

[3]The GPS coordinates are only available if a car is parked and available. There is no risk for users' privacy during rentals. In addition, no user's identifier is exposed. Therefore data is totally anonymized as there is no means to know who booked a car.

structure. The schema-less structure of MongoDB fits well in this work because it can handle the same collection documents defined with different key-value pairs. I decide to rely on such a system as I can easily manage the different field structures of providers, car2go and Enjoy in this use case. In addition, MongoDB offers great integration with Python through the *pymongo* module.

Four different collections compose the MongoDB data lake: *ActiveBookings*, *ActiveParkings*, *PermanentBookings*, and *PermanentParkings*. *ActiveBookings* and *ActiveParkings* are collections used to store information about the current status of cars (currently booked or parked respectively). These are temporary structures that make it easier to query each car's last observed status and update it. These are also instrumentals for real-time analysis of the system, e.g., to count how many cars are currently booked or available. *PermanentBookings* and *PermanentParkings* collections store the history of a past state of cars, for past bookings and parking, respectively.

For the documents in the bookings collections, I augment information by also inserting the expected route driving time and the public transportation duration on the same origin-destination pair. These two pieces of information are obtained through the Google Directions API using the initial and the final coordinates as an indication of the path.

The most important fields in the *ActiveBookings*, and the *PermanentBookings* collections are:

- *CarID*: the unique identifier of the car;
- *InitTime*: the initial time of the booking;
- *FinalTime*: the final time of the booking;
- *InitCoords*: the GPS coordinates of the booking star location, i.e., where the users picked up the car;
- *FinalCoords*: the GPS coordinates of the parking location where the car was dropped at the end of the booking;
- *DrivingTime*: The duration of the trip, expressed in seconds, as estimated by Google Directions API, following the best path;
- *PublicTransportTime*: The duration is expressed as arrival time of the best public transport trip, as estimated by Google Directions API, minus the *InitTime*;

Instead, the *ActiveParkings* and the *PermanentParkings* collections are characterized by the following fields:

- *CarID*: the unique identifier of the car
- *InitTime*: the initial time of the parking
- *FinalTime*: the final time of the parking
- *Coordinates*: the GPS coordinates of the parking spot

Algorithm 1: Data acquisition at time t

```
   Input  : t - Current timestamp
   Input  : S - Available Cars (crawling result)
1  AP = Read(ActiveParkings) // Get previous available cars
2  for car_j in S do
3      if (car_j in AP) then
4          del AP[car_j];
5      end
6      else
7          ActiveParkings.add(new Parking(car_j, t));
8          if (car_j in ActiveBookings) then
9              FinalCoords = car_j[coords];
10             ActiveBooking[car_j][FinalTime] = t;
11             InitCoords = ActiveBookings[car_j][InitCoords];
12             if (checkCarMovement(InitCoords,FinalCoords)) then
13                 ActiveBooking[car_j][driving_time] = GoogleApi(driving, InitCoords, FinalCoords);
14                 ActiveBooking[car_j][PublicTranportTime] = GoogleApi(public, InitCoords, FinalCoords);
15             end
16             MoveRow(car_j, ActiveBooking, PermanentBooking);
17         end
18     end
19 end
20 for car_j in AP do
21     ActiveParking[car_j][FinalTime] = t;
22     MoveRow(car_j, ActiveParking, PermanentParking);
23     ActiveBooking.add(new Booking(car_j, t));
24 end
```

Figure 2.2: Pseudocode of the data acquisition algorithm

I implemented an algorithm to extract booking and parking periods from snapshots, whose workflow is described in the pseudocode in Figure. 2.2. Here I describe each step.

I consider as inputs the snapshot S and the current timestamp t. Then I create a copy in the list AP of parked cars observed in the previous snapshot (as stored in the *ActiveParkings* collection) – line 1. I need the AP list to detect the cars that disappeared, i.e., have been booked at time t. This will be back explained later.

For each car car_j in the current snapshot S, I check if the car is present in the AP list. If so, it means that it did not change its status, i.e., it is still parked. Therefore, the car is removed from the AP list, and nothing is changed – lines 3-4. Otherwise, either the car has been parked in this snapshot, and the previous booking has finished, or the car is a new car added to the fleet. In both cases, a new parking starts, and I create a new document in the *ActiveParkings* collection – line 7. The *new Parkings* function creates a new document, sets the *InitTime* and *Coordinates* keys as current timestamp and car GPS coordinates.

I next check if car_j is present in the *ActiveBookings* collection. If so, the car was booked until the previous snapshot, and now it is back available. I thus finalize the previous booking and update its statistics. In particular, the toolsets the *FinalCoords* and *FinalTime* fields using the current car *coordinates* and timestamp – line 9-10. Next, I check if this booking includes an actual rental by checking if the initial position and final position differ – line 11-12. Recall indeed that customers may simply book a car but not finalize the rental. Specifically, Enjoy (car2go) offers a grace period of 15 (20) minutes, during which no charge is applied for a booking.

In case of an actual rental, I fetch the best path by i) car and ii) public transport from the *InitPosition* to the *FinalPosition* of the rental. I leverage the Google Directions API for this – line 13-14.[4] It is important to take into account that, while querying the public transportation time, the Google Directions API returns two pieces of information: how long the public transport takes to go from the initial to the final position and the estimated arrival time. It is fundamental to use this second information because it includes the time the user spends to reach the bus stop and wait for the bus. This is crucial, e.g., at night, when the first public transport solution may be available only several hours later.

After having processed all cars in the current snapshot, I iterate over the remaining cars in the *AP* list. Those are the ones that were present in the previous snapshot, but not in the current, i.e., the ones the new bookings. Finally, the software adds to the previous parking period by setting the *FinalTime* in the *ActiveParking* collection – line 21-22. At last, the tool creates a new booking via the *new booking* function – line 23.

I let *UMAP* scrape car2go's data from December 2016 to January 2018. In total it is possible to count about 27 million bookings spread in 23 cities. The same scraping procedure runs for Enjoy as well, from May 2017 to June 2019. The table 2.1 reports a brief resume of all the car2g bookings present in the data lake, while table 2.2 reports the same information for Enjoy.

2.4 Data Analysis

The third and final stage is the data analysis phase in which analytics modules query the MongoDB and obtain statistics. I rely on the Python programming language with Pandas and the GeoPandas libraries to deal with the data, the city zone definitions, provided by transport engineers as a shapefile, a popular geospatial vector data format, and the Geographical Information Systems (GIS) for the spatial analyses. I choose Python as it offers a large number of libraries that easily interact with different technologies like GIS, maps, and MongoDB. In particular, the usage of GeoPandas allows me to easily perform geographic analysis and split the city into many areas (or zones) of any possible shapes. I present more detailed characterization in chapter 3.

[4]https://enterprise.google.com/intl/it/maps/products/mapsapi.html

[5]wikipedia.org

Table 2.1: Overview of car2go's data sorted by number of bookings

City	City Size [km^2][5]	Population [5]	Avg. Fleat	Bookings
Columbus	583	892k	187	186k
Florence	102	372k	220	333k
Denver	401	727k	312	348k
Austin	704	964k	315	377k
Frankfurt	248	701k	242	505k
Toronto	630	3120k	400	536k
Amsterdam	219	854k	314	573k
Montreal	431	1704k	429	606k
New York City(Manhattan)	59	1629	500	739k
Turin	130	874k	396	868k
Munich	310	1464k	478	916k
Washington DC	177	705k	563	919k
Stuttgart	207	632k	486	1001k
Seattle	217	744k	710	1134k
Calgary	825	1239k	552	1176k
Rome	1287	2837k	582	1240k
Rheinland	-	1688k	648	1421k
Vienna	414	1915k	688	1702k
Madrid	604	3233k	424	2092k
Milan	181	1396k	776	2223k
Hamburg	755	1833k	812	2561k
Vancouver	115	631k	977	2701k
Berlin	891	3769k	1009	3091k

Table 2.2: Overview of Enjoy's data sorted by number of bookings

City	City Size [km^2][5]	Population [5]	Avg. Fleat	Bookings
Bologna	140	390k	72	47k
Catania	182	311k	78	205k
Florence	102	372k	82	279k
Turin	130	874k	251	983k
Rome	1287	2837k	601	2031k
Milan	181	1396k	755	3107k

2.5 Conclusions

In this chapter I described the software pipeline, named *UMAP* I used to harvest and store data from real FFCS providers.

The first stage explains the data structures and how I take snapshots of the system, getting all the cars ready for a ride.

The second step illustrates the algorithm that compares consecutive snapshots detecting car status variation. Here I introduced the two fundamental car status I use to describe each car history: *parkings* and *bookings*.

The third briefly opens several scenarios on the analyses of this data.

Chapter 3

Dataset Characterization

This work refers mostly to my paper " *UMAP: Urban mobility analysis platform to harvest car sharing data*, presented in at 2017 IEEE SmartWord conference [17]. My contribution is focused on data analyses and presentation.

3.1 Introduction

Mobility is one of the challenges to solve in our society and in cities, where eco-sustainability is becoming more and more important. Regulators and policymakers are positively looking into "smart" approaches to improve the current status of their urban network. The ability to inspect data is the first step to making informed decisions.

Car sharing refers to a car rental model where customers rent a car for a short period, usually for a few hours or less. One of its most exciting systems is the so-called *Free-Floating Car Sharing (FFCS)* system. This system's peculiarity is that customers can pick and drop the car wherever in a geo-fence area. The most famous company is car2go, which is present in 25 cities and 8 different countries, both in Europe and North America [1].

To rent a car in a modern FFCS system, users check on their smartphone or on the FFCS website which cars are available in the neighborhood. With a simple tap, they can book a car and start/end the rental. The FFCS app contacts a web-based backend server to fetch data about available vehicles, perform booking and accounting operations. Typically for this purpose, web APIs are used, some of which are publicly documented [18]. The same website and app offer information about the status of the car rental systems, and the same web API can be used to collect this information for free. In the past, this approach has been successfully used to obtain data for specific mobility studies – see section. 3.2 for more details. This chapter relies on the UMAP, the software described in

[1]The service is discontinued in North America since February, 29th 2020, https://www.share-now.com/ca/en/important-update/

chapter 2. In particular, I analyze the first chunk of data I collected. I let the crawlers run for 52 days, from December 10th, 2016 to January 31st, 2017. In total *UMAP* collected more than 104,000 *bookings* for car2go and 93,000 *bookings* and for Enjoy.

With these datasets, after a first cleaning phase where I detected entries corresponding to trips where the users made a ride, I characterize the FFCS service utilization to observe how people use these services, where they typically go, when, for how long the rental last, etc. Some observations are quite intuitive, e.g., people appear to be willing to use more the FFCS during weekdays and during peak-time. Counterintuitively, the rental duration and the driving distance show marginal changes over the day and weeks.

I complement the analysis by comparing the ride duration with the driving duration as suggested by Google Directions application, which UMAP can collect in real-time for each rental. This analysis highlights that 8.5% of bookings last less than the Google driving time. This may be due to Google Directions overestimating the driving duration or recalling that bookings include reservation time and the time to look for a parking spot. This may suggest that the time-based tariffs were adopted FFCS systems may encourage fast driving styles to reduce the rental cost. Next, I compare the booking duration with the equivalent trip duration by public transport as returned by Google Directions. I discover that rentals are 36% shorter on average than public transport time, but rentals start to be preferred when public transport time is higher than 10 minutes.

UMAP may represent an important support tool for the investigation of car-sharing users' habits. The scalable design of *UMAP* allows the policymaker to collect data from many FFCS providers and integrate it with other sources. This eases the analysis when taking into consideration trends and provider comparison. *UMAP* allows the Transportation Authority to make informed decisions when planning public transport systems. This characteristic strengthens the potentiality of *UMAP* for economic and sociological prediction and analysis. The data-driven approach, combined with other more traditional tools like surveys, represents an interesting observation point for understanding potential service improvements, both for car-sharing and public transport systems. The source code of *UMAP* is available for research purposes.[2]

The remainder of this chapter is structured as follows: section 3.2 discusses the related works. Section 3.3 presents the in the following order: first, car2go and Enjoy car usage over time characterization. Then, section 3.4 depicts how customers drive the cars and how they move in the city; finally, users' driving habits and the correlation between booking time and the public transport time. Section 3.5 concludes the chapter.

[2]`github.com/MobilityPolito/`

3.2 Related works

Since the diffusion of the new form of car-sharing based on a free-floating approach, many researchers from different fields have been dedicating increasing attention to these systems' analysis. The high demand for car-sharing has opened new challenges and perspectives in research.

One of the main topics is the study of fleet relocation policies [19, 20, 21]. On the one hand, concerning station-based car sharing, the free-floating system's flexibility may limit the operator's control over the drop-off zones. On the other hand, it allows for smarter strategies.

Herrmann, Schulte, and Voß [19] conducted a survey to understand how the availability of cars, and so the fleet relocation, affects the utilization of the service, and to develop and evaluate user-oriented relocation strategies. Those strategies were studied again by Schulte and Voß [20], who introduced an approach to support the decision of the vehicle relocation method to reduce costs and emissions in FFCS. Those kinds of investigations may result in instrumental support for the providers. In this direction, Wagner, Brandt and Neumann [21], analyzed the use of car sharing in Berlin, using indicators of attractiveness of specific areas, to develop a methodology that can help in business strategies, the expansion of the operative regions and to react to shifts in demand. In these works, the authors used data collected from car-sharing providers, using the car2go API [19, 20] or by a direct cooperation [21].

The study of the customers' behavior has been addressed by different researchers [22, 23, 24, 25, 26]. Schmöller et al. [22] studied factors that may influence car-sharing demand, carrying out an empirical analysis, considering FFCS in Berlin and Munich.

Kopp et al. [23] inspected the behavior of two categories of users, the members of an FFCS service (DriveNow), and the people who do not use car-sharing (NCS users), looking for different and distinctive mobility patterns. The impact of car sharing on people's mobility was addressed by Firnkorn [24], who proposed in his work a triangulation of two methods applied in the same survey to provide more precise measurements. Another approach was proposed by Ciari et al. [25], where a simulation tool, built on MATsim, an open-source project, was used to estimate travel demand for car sharing in the urban area of Zurich. An important question that can be addressed is how this new paradigm of transport is accessible to people. Tyndall [26] combined data of FFCS usage in ten US cites with demographic information, studying neighborhood infrastructures, population distribution, and their mobility habits. It has been showed that benefits of FFCS are distributed unequally, with a shift in usage in favor of advantaged populations.

Eco-sustainability is another essential asset for car-sharing services. Firnkorn and Müller [27] studied the environmental effects of FFCS in Ulm, registering lower pollution levels, and reducing private vehicle ownership.

This work aims to address all these challenges from the local administration's perspective to develop new transport and mobility policies. A study of this kind was recently conducted by Wang et al. [28] for the city of Seattle, where car2go was compared with public transport service. Kortum et al. [29] remark the necessity of using data-driven approaches to help decision making due to the lack of empirical data about free-floating car-sharing usage. They use a dataset obtained by InnoZ (Innovationszentrum für Mobilität und gesellschaftlichen Wandel) and containing the activity in 33 cities from 2011 to November 2015, to study the evolution in time of this mobility service. Those data, combined with demographic information, offered an aggregated point of view, over different cities, of the car-sharing service's growth and an understanding of the main characteristics. To the best of my knowledge, in the context of this case study, the only work on free-floating car-sharing was conducted by Ferrero et al. [30] from an economic point of view.

The majority of the previous works [19, 20, 21, 22, 23, 28, 29] leverage data collected in real-time or using surveys and interviews. Thanks to car2go APIs, which easily make available car-sharing data, a more data-driven approach is attractive for many researchers that start facing the problem of FFCS mobility analysis. Remarkably, only [29] seems to use data collected actively by different car-sharing providers. While authors use the information only for a specific purpose, i.e., analyzing the trend of car sharing through the years, I want to provide a broader perspective. The intent is indeed to offer a general-purpose methodology, both scalable and easy to interact with, to help researchers and local administrations analyze the mobility, harvesting data collected from FFCS platforms and other online systems, like mapping and direction services.

3.3 Temporal Analyses

In this section, I show several analyses to discover and characterize how the FFCS are used. In the first part of the section, I analyze the characterization of the temporal system to understand if FFCS are actually used and when.

I consider a period from December 10th 2016, to January 31st 2017, the first reliable collected data chunk. The system observed 125,000 snapshots, about 104,000 bookings for car2go and 93,000 for Enjoy. In Turin, the fleet of car2go was composed of 394 cars, and the fleet of Enjoy consisted of 172 cars.

In order to make clear the rest of the book, it necessary to univocally define the basilar entities related to the car status in the data lake described in the chapter 2, section 2.3.

Definition 1. *the **Parking** is the time period in which the car is present in, at least, two consecutive snapshots. Therefore, that car is available for a user reservation that may evolve in a user's ride.*

Definition 2. *the **Booking** is the time period in which the car is NOT present in, at least, two consecutive snapshots. Therefore a user booked the vehicle, or the provider made it temporarily unavailable for maintenance.*

3.3.1 System Utilization

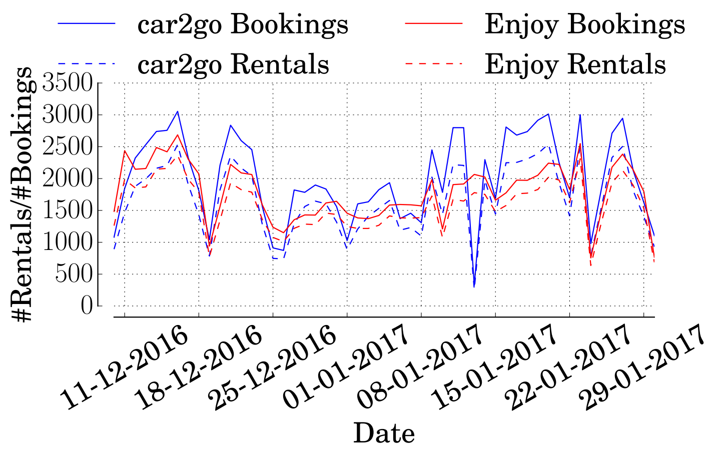


Figure 3.1: Total number of bookings and of rentals per day for car2go and for Enjoy

The providers, in this case, study, allows the users to *reserve* a car before the ride. More in detail, the provider makes the reserved car unavailable for the other users without billing the customer who reserved the car. When the reservation time (that changes for each provider), the billing mechanism starts even if the engine is still off. The customer can cancel the reservation without any expense if it happens before the reservation time. With this mechanism, the providers would let the possibility to the users to reach the cars by foot.

Given that, it is now possible to define:

Definition 3. ***Reservation*** *A reservation is a booking where the initial and final destination matches and the duration is lower than the provider's reservation time.*

Definition 4. ***Rental*** *A rental it is a booking where the initial and final destination are different.*

Starting from December 10th, figure 3.1 plots the total number of bookings and the total number of rentals recorded on each day, for car2go (blue curves) and for Enjoy (red curves). Obviously, being the latter a subset of the first, its number is always smaller. However, during some days, the discrepancy is well visible; that means that the operation of booking cancellation is not so rare.

Interestingly, firstly, both car2go and Enjoy follow a similar behavior with the number of bookings and rentals decreasing in the Christmas period and increasing again after the Epiphany.

Secondly, despite the fact that the car2go fleet has more than twice as many cars as Enjoy (394 vs. 172), the number of car2go bookings does not show such a higher value with respect to Enjoy. Enjoy having more bookings in some snapshots, e.g., December

10th and 11th. Moreover, at some points (December 19th, January 24th), it is possible to detect a huge drop due to the crawler failures.

Moreover, some drops in bookings' values are noticeable. Those sudden changes can be addressed to some failures, in the crawlers (e.g., when all curves suddenly drop) or in the operators' web services(e.g., when only one system suffers a sudden drop).

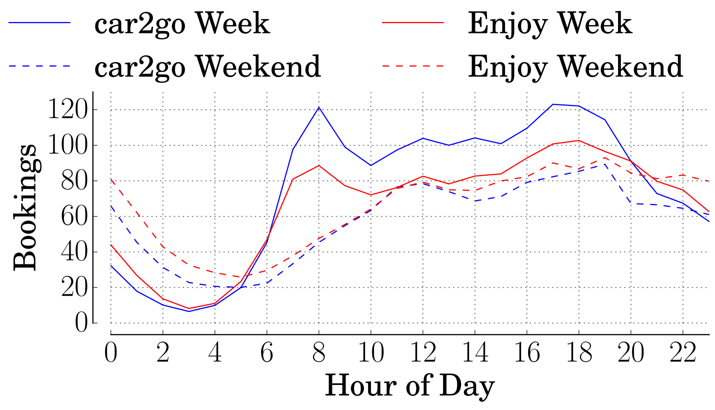


Figure 3.2: Mean number of bookings in weekdays and weekends for car2go and Enjoy

Looking at the data with finer granularity, it is noticeable that the car-sharing adoption changes during the day. To better characterize this, I separate weekdays and weekends. The figure 3.2 points out the trend over the day. The curves report the average number of bookings over the entire period in each hour of the day.

Firstly, it is possible to see that weekdays and weekends have a quite different trend. During the weekend, FFCS systems are more used at night with respect than weekdays, with on average at midnight of 80 and 60 bookings per hour for Enjoy and car2go. Instead, the figure shows how during the weekdays, both car2go and Enjoy have their peak of usage at 8 am and between 5 pm and 7 pm. This trend can be easily explained as, during that time slots, FFCS customers use cars to go and return from work. As previously indicated, despite car2go has twice the number of cars than Enjoy, the system utilization of the latter is higher, with peak utilization topping 60%, versus 30% of car2go.

Even in the absolute number of rentals, Enjoy shows a higher number of bookings after 8 pm during the weekdays, and always during the weekends. This can be explained by the car models adopted by the two companies. While car2go uses the compact-two seats *Smart*, Enjoy fleet is composed of *Fiat 500*, which are four-seat cars. Rentals prices are instead comparable (0.24€/min Enjoy vs 0.25€/min car2go). Data suggests that Enjoy looks more appealing during the times when people prefer to share the ride and during weekends when families and groups move.

3.4 Rides characterization

In this section, I give a detailed look at driving habits. In particular, I compare driving distances versus rentals and parking duration. Finally, I conclude with some insights about the variation of spatial demand, characterizing which and because some zones attract or generate more rentals with compared to another one.

3.4.1 Driving patterns

jNow, I show how users tend to use FFCS systems during weekdays and weekends. I study three different aspects of users' behaviour:

- for how long users reserve the car before cancelling a booking (figure 3.3)
- for how long users rent a car (figure 3.4)
- how far users drive (figure 3.5)

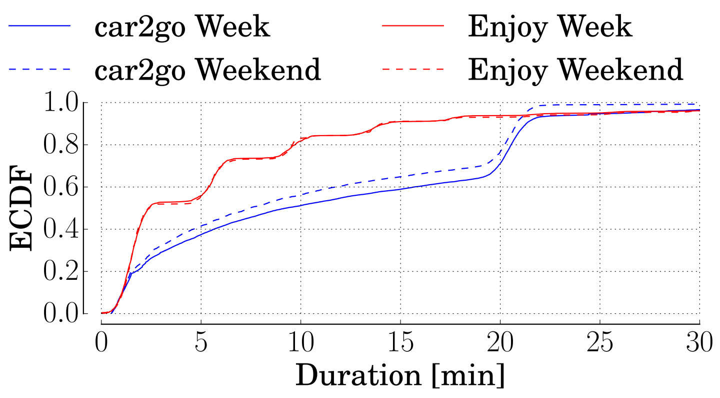


Figure 3.3: ECDF of the booking duration when the booking does not produce a rental. Weekdays and weekends

First, I check if and for how long users reserve a car, and they cancel a booking. Interestingly, only a small subset of Enjoy bookings are affected by cancellation with respect to car2go bookings. In particular, the dataset presents 14.9% of car2go and 2.9% of Enjoy bookings cancellation. This again hints at people preferring to use the Fiat 500 offered by Enjoy, so that they hardly cancel a booking when they reserved an available vehicle. On the contrary, the car2go cars' availability is higher, and so it looks more comfortable to find a closer car. People may thus cancel a previous booking when they see a closer vehicle. Another hypothesis is that car2go may be used as a "backup" until an Enjoy vehicle becomes free in the user's area.

Looking at when people cancel the reservation, figure 3.3 shows the CDF of reservation time. Indeed, car2go tends to have a smaller percentage of cancellation within 5 minutes, with a massive step at about 20 minutes. While the first ramp can be explained as a communication error or as some sudden cancellation, the latter can be explained by the *maximum free-of-charge reservation time* of car2go. Indeed, users may reserve a car for up to 20 minutes without paying any fee. The same trend is not present for Enjoy, which offers a *maximum free-of-charge reservation time* of 15 minutes. Instead, the curve shows a peak at 2 minutes and then a decreasing trend after 15 minutes, when almost all the cancellations are already made. One last important aspect that this picture shows is how the Enjoy curves have some steps instead of being as smooth as the car2go ones. This hints at periodic updates on the web system so that a time granularity emerges. To shed some light on this phenomenon, I performed some active experiments with the Enjoy web portal. The experiment consists of making a new reservation and find when the crawler detects that the car actually disappears from the set of available vehicles. Then, as soon as I spot the car disappearing, I cancel the reservation to detect when the car reappears in the system. Surprisingly, I discover that when the users make the reservation, the car immediately disappears from the system. Instead, when I cancel the reservation, the system takes between 1 and 4 minutes to actually show the car again. The presence of such an offset causes the steps in the Enjoy curves, which are affected by an artificial delay. To take into account, this offset all Enjoy duration have been decreased by 2.5 minutes, i.e., the average delay the Enjoy system adds.

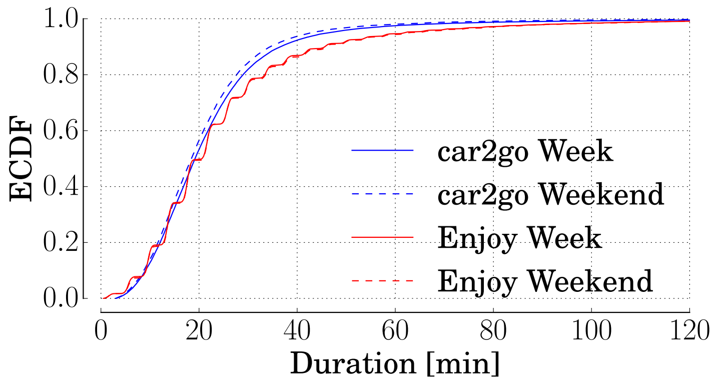


Figure 3.4: ECDF of the rental duration. Weekdays and weekends

I next move to characterize the rental duration. Figure 3.4 depicts the Empirical Cumulative Distribution Function (ECDF) of the booking duration for Enjoy and car2go during the weekdays and the weekends. The plot shows how the trend tends to be equal during the weekdays and the weekends. This demonstrates that, despite the different pattern of utilization shown before, the booking duration time results similar. Secondly, the ECDFs of car2go and Enjoy are almost overlapped, highlighting how these two services tend to be used in a similar way. Indeed, most of the rentals last less than 1 hour, with 80% of them lasting less than 30 minutes. It is important to remark that these times also include the reservation time, i.e., the time the user can reserve a car for free before driving it, and the time to find a parking place. Therefore the actual driving time may be significantly smaller.

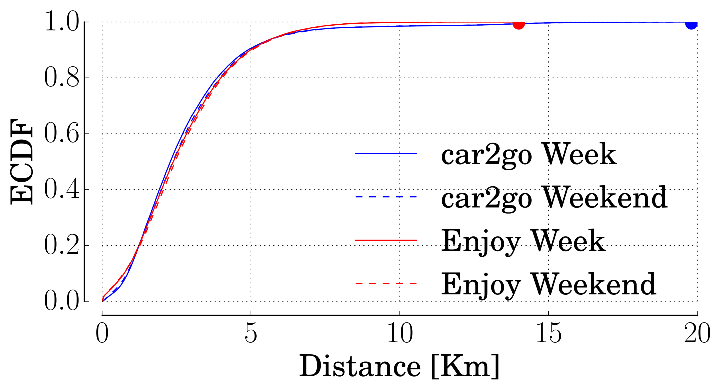


Figure 3.5: ECDF of the rental driven distance. Weekdays and weekends

I repeat the same analysis, considering the driving distance, as reported in Figure 3.5. To determine the driving distance of each trip, I exploit the Google Direction APIs to get the shortest path from the origin to the destination. Similarly, for the driving duration, car2go and Enjoy show comparable behavior and marginal changes during weekdays and weekends. Interestingly, the graphs point out that 90% of the trips last less than 5 km, demonstrating that most of the rentals are used for short trips both in terms of time and in terms of distance. Lastly, the car2go curves saturate many km later than the Enjoy ones, as highlighted by the circles. This is due to the possibility to reach the airport of Turin with the car2go cars, which is about 20 km far.

3.4.2 Spatial Analysis

In the previous section, I analyzed car2go and Enjoy only from a temporal point of view. In order to have a complete scenario, it is necessary to study recurrent spatial patterns. To do that, I projected the initial and final coordinates on the Turin's neighbor map. Then I computed the attractiveness of each neighbor. Figure 3.6 shows the attractiveness of the zones in Turin by analyzing the departure and arrival zones. For each zone, I compute the difference between bookings ended in the evening [5 pm - 9 pm] and bookings ended in the morning [7 am, 12 am]. Red areas are those more attractive during the evening, while blue areas are more attractive in the morning. It is clear that the city center is the most popular destination for car-sharing during office hours, while the trips are sparsely ending in the suburbs during the evening.

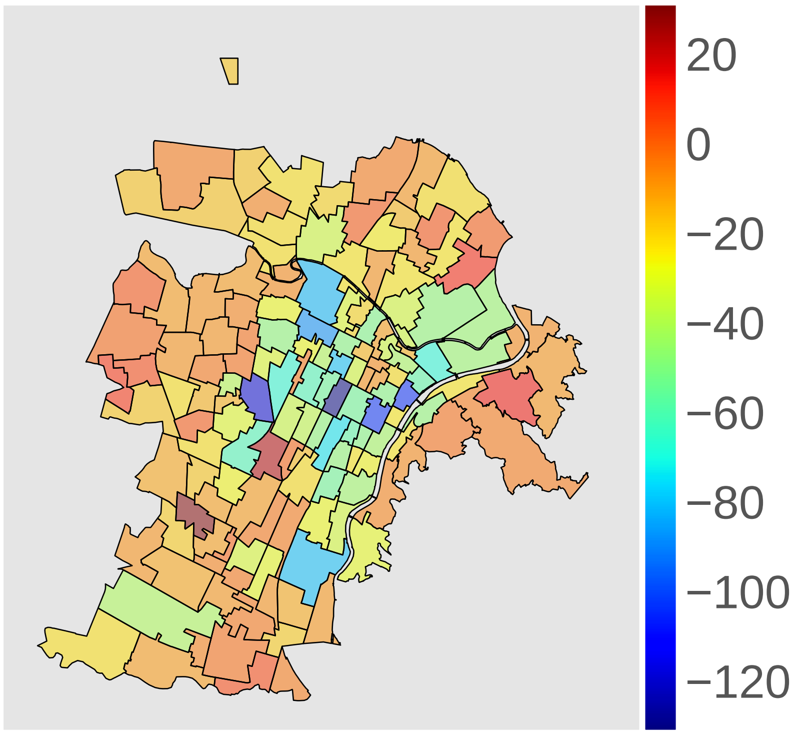
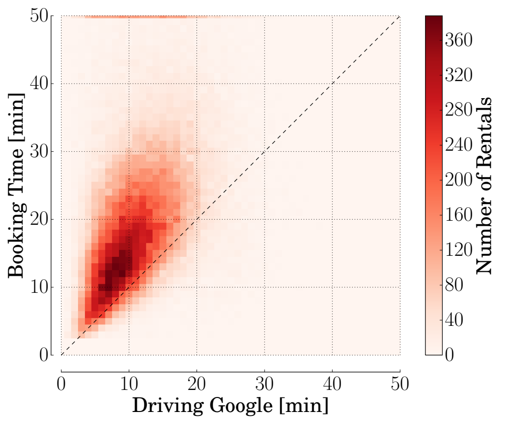


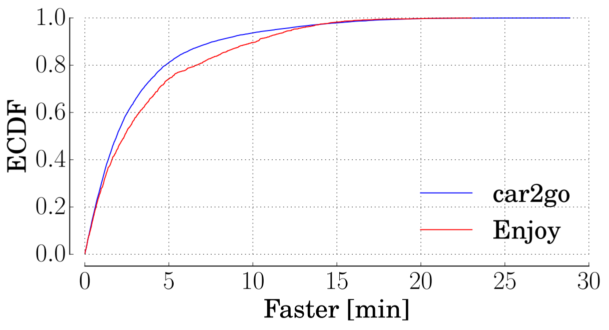
Figure 3.6: Heatmap of arrival - departure per area from 7 am to 12 am vs from 5 pm to 9 pm

3.4.3 Users' Habits

I now characterize how users drive and what is the correlation between public transport usage and availability.

To observe users' driving habits, I use the *driving time* returned by the Google Directions APIs to obtain the estimated driving time from initial rental position to the final rental position. Intuitively, the rental time is longer than the driving time as it takes into account also the reservation time and the time to find a final parking spot. Figure 3.7a shows a heat map where the X-axis represents the Google estimated driving time and the Y-axis the actual booking time. Each cell counts the number of observed trips for each (x,y) pairs. For ease of representation, the values are rounded by the minute. The diagonal line separates the area where the booking time is lower/greater than the driving time. As expected, most of the trip falls in the area where the booking time is greater than the driving time. However, a nonnegligible number of trips (12.1%) falls in the area where the booking time lasts less than the driving time.





(a) Heatmap of booking time vs estimated driving time by Google

(b) ECDF of the difference between the expected driving time and the actual driving time

Figure 3.7: Users' driving habits

This may be due to several factors: Google Directions possibly overestimating the average trip duration, or users driving faster than expected. To better quantify how much faster users drive the car in those cases, I computed the difference between the driving time and the actual booking time. I show the Empirical Cumulative Distribution Function of such values in figure 3.7b. It is possible to see that most of these trips are only 5 minutes faster than the estimated driving time, with Enjoy users who seem to drive faster than car2go ones. Indeed, if the trip is more than 10 minutes faster, Google suggested a longer path to the destination, e.g., suggesting to take the highway, which was much longer with respect to crossing part of the city.

This analysis hints that the current pricing policy, which depends only on the booking time, may have some drawbacks as it may encourage users to drive fast. A hybrid pricing policy, which takes into account both the time and the distance, may be effective in solving this problem, e.g., by increasing the price in case of a user drive faster than expected or by reducing the fee in case of traffic congestion.

Another aspect that Google Direction data can help to resolve is to estimate the actual traveled distance. In particular, the system computes the Euclidean / Haversine [3] distance between starting and final position to which I add the Google Direction path estimation (see section 2.3 for more details) for each ride present in the dataset. Given that it is possible to track in real-time cars, I study how to derive the total amount of traveled kilometers correlating only the straight and Google distances.

Figure 3.8 reports the ECDF of the ratio between the Google Distance and the straight distance. The plot shows how, in the 95% of rides, the Google distance is about twice

[3]https://en.wikipedia.org/wiki/Haversine_formula#cite_note-Brummelen_2013-1

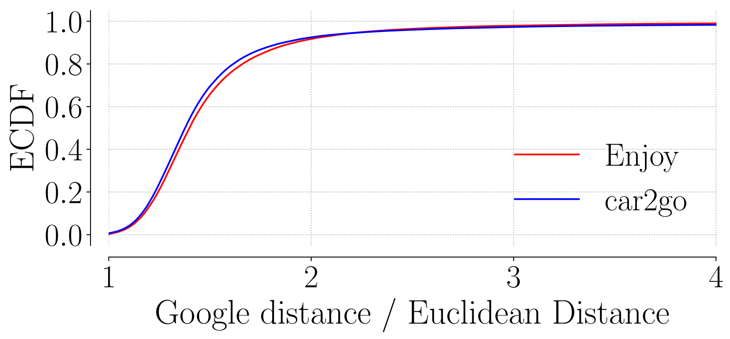


Figure 3.8: ECDF of the ratio between the Google Distance and the straight rental distance

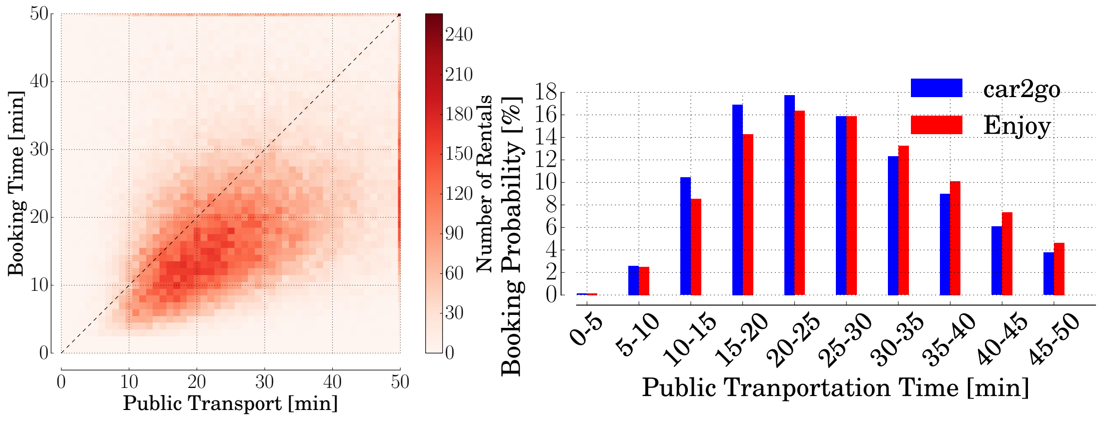
time as long as the straight distance. However, on average, it is possible to observe a multiplicative factor of 1.5 for and enjoy. Anyway, in the following chapters and in particular in chapters 5, 6 and 7 I used the median value, of 1,4. Notice that the bigger values in car2go are due to a bigger operative area. Table 3.1 resumes the main ECDF values.

Table 3.1: Statistics on corrective factor for car2go and Enjoy

Provider	**Mean**	**Std**	**Min**	**25%**	**50%**	**75%**	**90%**	**95%**	**99%**
car2go	1.5	0.54	0.53	1.27	1.40	1.59	1.90	2.21	3.47
Enjoy	1.48	0.52	0.87	1.26	1.37	1.54	1.81	2.12	3.38

At last, I leverage Google Directions APIs to extract public transport travel information for each vehicle's trip. I want to analyze another way of mobility in the urban area and compare car sharing usage with respect to public transport. Results are shown in figure 3.9. As one could expect, the majority of trips last less than public transport. The higher density is for bookings that last between 10 and 20 minutes. For longer trips, the discrepancy in terms of duration is higher, probably due to the longer path and the higher number of stops of public transport. Conversely, I can interpret the points where the booking time is greater than the public transport duration as trips where the customers spent a lot of time reaching the car or finding a parking spot for the drop-off.

To help to visualize the juxtaposition of car-sharing and public transport, I extract from the data the probability of booking a car, conditioned to the public transport travel time. Figure 3.9b, reports on the X-axis the public transport duration (as predicted by Google) in intervals of 5 minutes, and on the Y-axis, the probability of booking a car for each interval. The distribution of probability is similar for both car2go and Enjoy. Higher values are reported for trips that can be covered by public transport between 15 and 35 minutes. Interestingly, car-sharing mobility is not preferred for very short trips, while the distribution shows a significant tail for a duration greater than 30 minutes.



(a) heatmap of the booking time vs the estimated public tranport time by Google (b) Probability of using car sharing based on public transport time by Google

Figure 3.9: Public transportation vs car sharing

This behavior can be justified by the significant amount of time that can be saved with cars-haring with respect to public transport.

Finally, to globally understand how users tend to use the different services, I report in figure 3.10 the average time for the Enjoy rentals (red curve), the car2go bookings (blue curve), the driving time (green curve), and public transport time (orange curve). To compute this value, for each hour, I take all the rentals of interest, and then I compute the average value and report it. The first interesting aspect is that the average time of Enjoy is always greater than the car2go ones and for the pure driving time. Secondly, both show a similar trend with a decreasing average duration during the night. As a consequence, it is unjustifiable to ascribe this trend to traffic jams, instead, but more likely with an increased time in the reservation time and in the parking time. Finally, it is possible to appreciate how during the night the public transport takes more than 1 hour for trips which last less than 20 minutes by car. Instead, during the daytime, the average public transport time gets close to the car-sharing time.

3.5 Conclusions

In this chapter, I presented some analyses made possible through *UMAP*, a platform I designed and described in chapter 2 to collect and store data, and able to extract higher level information from FFCS provider.

By analysing the data, I highlighted different aspects related to the system utilization, how users move in the city in different periods of the day, and what are the users' driving habits.

This study points out the significant amount of information that it is possible to extract from well designed data collection pipelines. More in details, by looking the

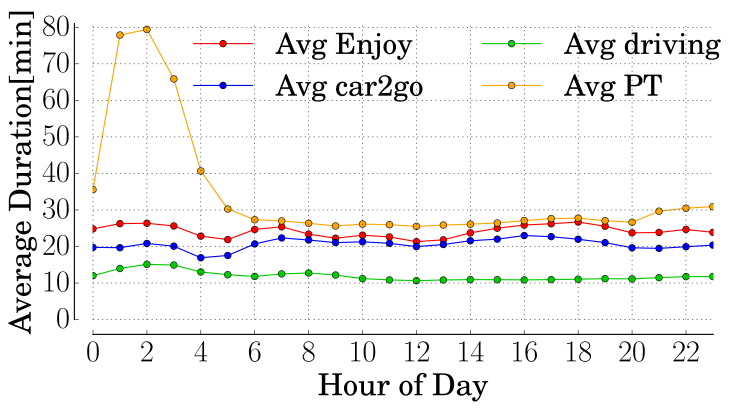


Figure 3.10: Average Time per transport solution per hour

system utilization, I demonstrated that FFCS cars are frequently used for short trips which last less then 30 minutes and 5 km. Moreover, despite Enjoy has a smaller fleet, its system utilization is frequently higher than car2go one due to the more appreciated car model it offers. Exploiting the spatial analysis, I highlighted how users tend to move during different time periods. Finally, the users' driving habits showed us that current charging policy may encourage users to drive fast.

Chapter 4

Characterizing Client Usage Patterns and Service Demand for Car-Sharing Systems

This chapter refers mostly the paper *Characterizing client usage patterns and service demand for car-sharing systems* [31], published on the Journal Information Systems, available online since October 11, 2019. My contribution is mainly focused in all the aspects about car2go analyses.

4.1 Introduction

Urban mobility is a key research area, attracting several academic studies and private investments. It is intrinsically connected to a wide number of urban activities, such as the demand for communication resources. Understanding urban mobility, specifically the traffic-related mobility with motorized vehicles, is currently acquiring more importance in order to improve the people's quality of life studying, for example, road mesh planning and communication resources allocation [32, 33].

The first step in understanding urban mobility patterns is the proper acquisition of data. Data can be obtained in several ways, e.g., by observing vehicles passing through sensors or fixed/mobile radars, acquiring traffic data from cameras, or by the active participation of users (*crowdsourcing*). However, large and heterogeneous data acquisition is still a challenge. Indeed, only a few companies have access to them, and usually, they embed some random components to protect the users' privacy [17]. Therefore, it is important to collect and study the available open data and generate models that can help understand urban mobility and people's social interactions in the urban environment.

Many alternative transport modes contribute to urban mobility. Among them, the car-sharing paradigm is quickly growing [34, 17, 35]. In a car-sharing system, people can drive a vehicle without worrying about buying it and paying for maintenance, fuel, and parking fees. By 2015, more than 1.5 million users and 22 000 shared vehicles have been counted in the Americas, and growth in usage is still expected [36]. Overall, car-sharing services are classified into three categories:

- the one-way services, where the vehicles are available in specific stations and the user can move a car from a station to another;
- the two-way services, where the user must return the vehicle to the same station she/he picked up the vehicle;
- the free-floating service where vehicles are not tied to stations. In this case, the users are able to start and finish their trips everywhere within an operative area and in public parking spots [34];

.

This chapter proposes a comparison between free-floating and different station-based car-sharing paradigms. More in detail, we characterized those services in order to outstanding different users' habits. We take a case of study the city of Vancouver that hosts several car-sharing providers. Our characterization relies on data we gathered for more than a year from Modo, Evo, and car2go [1] car-sharing services —a two-way, a one-way, and a free-floating service, respectively—. The chapter illustrates the users' demand and usage patterns of vehicles from these services and, at a glance, the contributions are twofold: first, a characterization of three important car-sharing paradigms and, second, a demand model for their vehicles, providing statistical distributions which describe their busy and idle periods. This study is important to highlight particular situations where car-sharing services are attractive and, together with data from other transport modes, to uncover trends and mobility patterns. Moreover, we also believe the collected data and the developed models can be used to generate an accurate synthetic workload. Consequently, these can contribute to the development of better capacity planning models for car-sharing systems and a better plan of public transport systems.

The remainder of this chapter is structured as follows: section 4.2 describes related work; section 4.3 describes details of the three car-sharing paradigms; section 4.4 discusses the data collection and analysis methodology for all services; section 4.5 presents the results of the characterization for each model and the comparison of them, whereas section 4.6 concludes the chapter.

[1]service dismissed on February 29, 2020

4.2 Related Works

Prior works on one-way car-sharing services revealed some important characteristics of these services as its usage patterns and their impact on the urban centers [35, 37, 38, 34]. For example, one-way car-sharing systems are mostly used in dense urban areas with good public transportation system [39]. Young people with a higher education level are more attracted to use this service [40]. Moreover, several works also confirm positive impacts on the actual transport system, such as the reduction on traffic and emission of pollutants [41, 38], the increase of free parking spots and in the use of public transport [42]. These prior works also reveal that one-way car-sharing services are used for long journeys and shopping [37]. In most cases, at least two passengers use the vehicle [35]. Finally, these works also reveal interesting features about the fleet of electric cars. For instance, vehicles remain parked in central regions for lower periods than in suburban regions, directly impacting the autonomy of the vehicles [34].

Previous works also point out the differences between the free-floating and the one-way model services. Indeed, the free-floating vehicles are often used for shorter periods, presenting commuting trips and a considerable number of trips to airports [37], [35] [43]. Typically, free-floating vehicles carry a single user [35] and this user presents fast driving habits [17]. Finally, the free-floating model also presents a periodical usage: during the mornings, central areas of the city are the main destination, while during the evening, suburban areas are reached more [17]. Despite the flexibility of the free-floating and one-way model, previous works have not observed a clear difference in users' preferences between them [37]. On the other hand, some works have identified that these services attract different user classes, exposing the fact that free-floating models and station-based models must be treated separately [35].

To the best of our knowledge, only our prior works characterize the two-way car-sharing service model [44, 45]. More precisely, in [44] we first characterize the usage patterns and the demands of *Modo*,[2] a car-sharing service that operates in Vancouver (Canada) and nearby regions. We present a simple model that represents the demand for vehicles in this car-sharing system, presenting statistical analysis to parametrize this model. Then, in [45], we further explore this two-way car-sharing service model by evaluating two distinct periods and also present a spatial analysis of the vehicle demands. Our results evidence long travel duration and many cancellations, which produce a low utilization factor of the system. Moreover, the two-way system usage presents a strong relationship with the public transport system, as well as with regions nearby points of interests, such as public universities and commercial centers [45]. In [46, 43] we analyzed free-floating car-sharing data in different cities and propose models and optimization methods in order to efficiently use electric cars.

[2]http://www.modo.coop/

4.3 Car-sharing systems

The car-sharing systems might be implemented according to two paradigms: station based and free-floating. The station-based paradigm can be divided into *one-way services* and *two-way services.* Both Station-based models require that a user picks up the vehicle she/he will use at a defined parking spot, i.e., charging station for electric vehicles or providers' parking spots. The user, in turn, may leave the vehicle at any of the base stations scattered throughout the service coverage region (i.e., one-way car-sharing service), or she/he may be obliged to return the vehicle to the station of origin (i.e., two-way car-sharing service). On the one hand, the two-way model requires simpler logistics and infrastructure compared to other models. Its implementation can be performed faster and at a lower cost. On the other hand, the one-way model may be more flexible and cost-efficient to users than a classical rental. For example, in case there is a base station near the final user destination, she/he may leave the car at the station while performing other tasks. The time the vehicle is parked is not charged, incurring lower costs to users. However, a parked vehicle may be reserved by another user. The free-floating model does not require any fixed station. In other words, users reserve a car, parked in non-reserved spots in the streets. By the end of the use, users may leave vehicles at any location in a predefined area. Notably, the free-floating model eliminates the limitations that station-based models hold, making the experience more flexible and closer to private-owned vehicles [37].

Figure 4.1 presents an abstract model that describes the possible states of a vehicle in any of the three car-sharing systems. Intuitively, a car can be in use (i.e., *busy*) or *idle.* A *busy* vehicle is *rented*, meaning that someone is paying for it during this period. On the other hand, *idle* vehicles may be *unavailable* (i.e., during a maintenance process), *available*, which means someone can reserve or it, or *reserved.*

The state in which the car is ready for a customer is *available.* In this situation, the user can reserve the car and subsequently begins the ride or start to drive the vehicle instantaneously. In the first case the state changes from *reserved* and then *rented* while in the second case the state switches into *rented* directly. When the customer concludes the rent the vehicle state moves from *rented* to *available* returning ready for another rent. Notice that if a user reserves the car and then cancels the reservation, the vehicle state moves from *reserved* to *available* without assuming the state *rented.* If a vehicle is not in one of the previous three states, then it is *unavailable*, e.g., it is out of service. As we will see in the next Section, not always the data contains plain information about which of the four states the vehicle is. We will need to infer it by making some assumptions deducing the car state by filtering the rentals according to the duration and the possible driven distance.

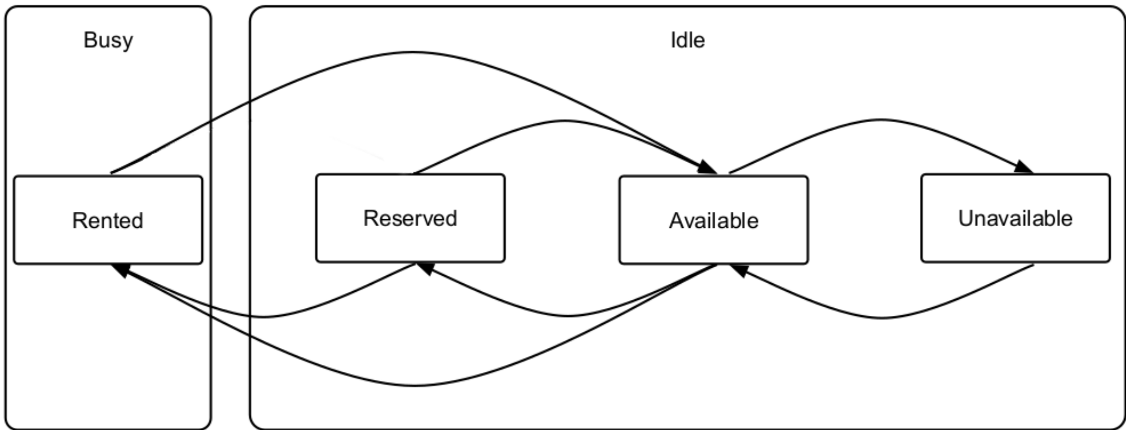


Figure 4.1: Possible states of a vehicle in a car-sharing system.

4.4 Datasets and crawling methodology

This work relies on usage data from three car-sharing services: Modo, Car2Go, and Evo. These services operate in several cities and countries. We focus on data from the Vancouver area, where all these three services operate. Modo fleet is composed of combustion, electric, and hybrid cars; Car2go offers combustion cars, and finally, Evo supplies only hybrid vehicles. For each service, we collected both users' trips and fleet composition. In total, we observed more than 680 cars for Modo, 1 200 for Car2go, and 1 000 for Evo.

For all the three services, we collected vehicle status minute-by-minute, through public Application Programming Interfaces (APIs) or, directing accessing their service information web-page. From those requests, it is possible to scrape also a unique ID and position of each car present in the dataset. In short, through the Modo API[3] returns the station and vehicle IDs. Modo provides vehicle status, too: a car indeed may be available, reserved, or running. Evo's data[4] information page allow to check the remaining vehicle fuel (in percentage) and its location. Finally, Car2go APIs[5]. Data from Evo and Modo comprises five months, ranging from March 1st, 2018 to July 16th, 2018. Car2Go data comprises thirteen months, ranging from December 31st, 2016 to January 31st, 2018. It is important to notice that, to not violate the users' privacy, the providers do not expose any users' personal information. Moreover, the companies do not track the cars during a trip, so we do not know exactly the travel path, but only the start/end positions and the duration of travel.

All measurements used in our analyses are publicly available in the following trace

[3]Modo API, `http://modo.coop/api/`

[4]Evo public portal, `https://www.evo.ca/api/Cars.aspx`

[5]Car2go API, `https://www.car2go.com/api/tou.htm`, last access February 2018

repository: `http://netlab.ice.ufjf.br/index.php/carsharingdata/`

4.4.1 Modo crawling methodology and data summary

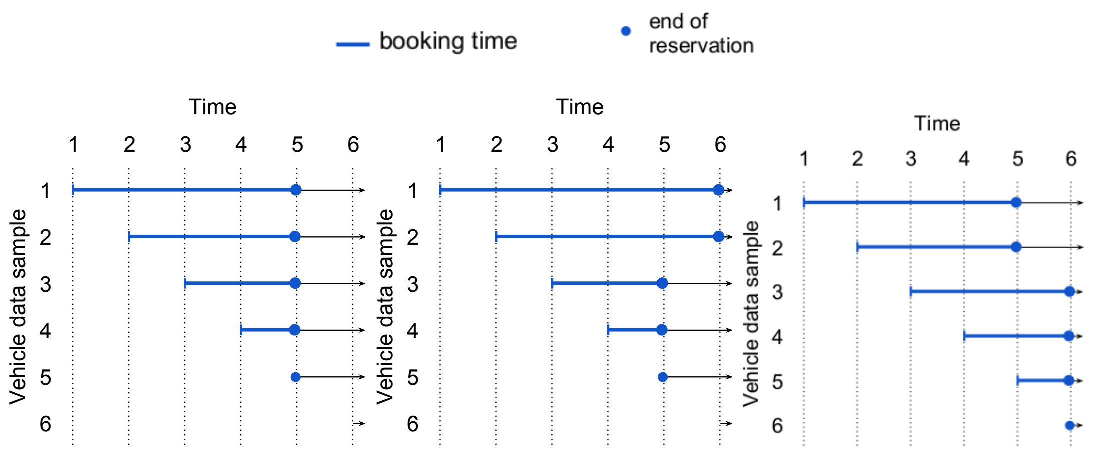
The Modo data collection process was conducted with a crawler that uses its public API. First, we request to the Modo API the list of all vehicles of the service. Then, minute by minute, we request the status of each of these vehicles. Each request returns the schedule of a vehicle, informing the periods it will be available for the next 24-hours. Moreover, it returns the vehicle location, i.e., the station, with its identifier. Note that Modo API does not return specific vehicle status, nor any information that could be used to identify users of the system. We uncover if a vehicle is busy or idle based on its reservation period and the current observation time. In other words, we collect several vehicle schedules and compare each other. Figure 4.2 illustrates the process of collecting data for a given vehicle. Each data sample corresponds to a request to the API in the order they occur. Data sample #1 is the result of the API request at minute 1 ($t = 1$), data sample #2 is the result of the API request at minute 2 ($t = 2$), etc. At each data sample, the blue dot represents the time a vehicle will be available. We highlight three possible situations:

- First, as shown in figure 4.2(a), at $t = 1$ a given vehicle is shown reserved up to $t = 5$. At $t = 2$, the new request to the Modo API still show us that the vehicle will be available only at $t = 5$. Each of the following requests to the API confirms the booking period. At the time $t = 6$, we perform a request to the API, and the vehicle is no longer booked. In sum, we are able to infer that someone booked the vehicle before or at $t = 1$, and returned it to the station at $t = 5$.

- Second, as shown in figure 4.2(b), at $t = 1$ the Modo API returns that a given vehicle is reserved up to $t = 6$. However, in this case, a request at $t = 5$ shows the vehicle is no longer reserved. In this case, we can infer that the user returned the vehicle earlier to the station, which means she/he used the vehicle only up to $t = 5$.

- Finally, as shown in figure 4.2(c), the user may extend the booking period. More precisely, at $t = 1$ the given vehicle is reserved up to $t = 5$. At the third request, we note that the vehicle will no longer be available at $t = 5$ but $t = 6$. The following API requests confirm the use of the car until $t = 6$.

Besides, we also collect base stations location, vehicle models, and whether the vehicle is electric or hybrid. Table 4.1 summarizes the data we have collected from Modo. We stored 134 million records in 5 months, from a fleet of 682 vehicles distributed in 528 stations, each of them with one or more cars. The stations are located in Vancouver, Canada, and neighboring cities. This data allows us to analyze more than 98 000



(a) Normal situation (b) Anticipated vehicle return (c) Extended travel situation

Figure 4.2: Possible vehicle status during the Modo crawling. In (a) a normal booking and usage situation; (b) a cancellation situation; (c) a consecutive booking situation.

travels.[6]

# of Collected Records		≈ 134 000 000
# of Booking Records		149 732
# of Travels Records		98 915
# of Stations		528
	- Common	530
# of Vehicles	- Hybrids	148
	- Electrical	4

Table 4.1: Summary of the Modo dataset.

4.4.2 Evo crawling methodology and data summary

Evo does not offer a public API to researchers. For this reason, we collect data that is publicly available at its web portal. Minute by minute, we retrieve a list of all system vehicles. Moreover, we request service snapshots describing which vehicles are parked, where they are parked, and if they are available to travel. We process all snapshots of the system to infer the moments a vehicle is busy (rented) or idle (parked at a station). During a snapshot, if a vehicle is listed among the system vehicles, but it is not parked at any station, we infer it is in use. Then, we set-up the travel starting point as the last

[6]Data are available on http://netlab.ice.ufjf.br/index.php/carsharingdata/

station the vehicle was parked. Analogously, the travel ending point will be the next station the vehicle appears in a future snapshot. The total travel time is accounted for as the difference between these snapshots times. For each travel we identify, we also record the end-to-end path, according to the Google Maps API. In this way, we are also able to calculate the estimated travel, taking into account the local traffic conditions. Clearly, this estimation does not take into account the car-sharing client behavior and, as a consequence, differs from the real travel time we also store. One may reserve a car in Evo and cancel this reservation, within a thirty minutes range, without any charges. Thus, we infer the number of cancellations in Evo by filtering short travels (i.e., < 30 minutes) where the start and endpoints are the same. To accommodate GPS imprecision, we consider a 3 meters threshold. Table 4.2 summarizes the data we collect from Evo. Note that this service does not need a large number of stations because the user can park the car in some public park spots in the service area that is called home zone (Vancouver and neighboring cities).

# of Collected Records	142 853 500
# of Travels Records	644 887
# of Stations	130
# of Vehicles	1 237

Table 4.2: Summary of the Evo data collection.

4.4.3 Car2Go crawling methodology and data summary

The car2go data collection is widely described in chapter 2.

Recalling that the software is able to detect two events, corresponding to the car status change, clearly observable from the data. Indeed considering the current time instant t_i:

- if in t_i the car is present in the API reply and at time t_{i+1} it is not, that car passes from available to rent.
- if in t_i the car is *not* present and at time t_{i+1} it reappears in the API reply, that car passes from rented to available. It represents a booking finish and a parking beginning. For privacy constraints, the position of the car during a booking is not available.

Moreover, the straight distance (computed with the Haversine formula) between the ride starting and final points is not the real driven distance hidden for privacy issues. For this reason, the software attaches for each ride the distance provided by the Google Maps API in order to have a better estimation of the driven pattern.

Table 4.3 summarizes Car2go dataset used in this chapter. We have more than one million travels in thirteen months of data. As a free-floating service, Car2Go does not have stations, but it has an operation zone that covers a large area of Vancouver city and North-Vancouver.

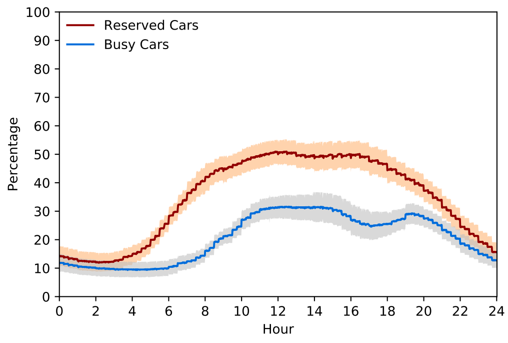
# of Travels Records	1 095 577
# of Vehicles	1 077

Table 4.3: Summary of the Car2Go data collection.

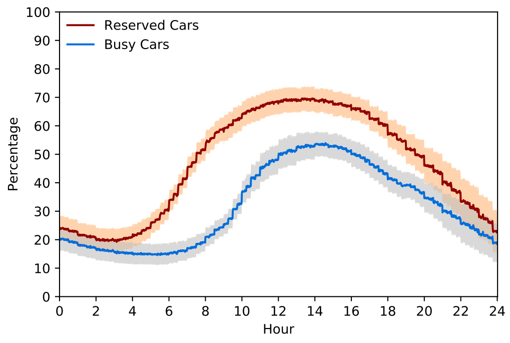
4.5 Car-sharing services characterization

In this section, we first present temporal characterization of the three services (section 4.5.1). Then, we describe the services spatial-temporal characteristics (section 4.5.2). Finally, we present users' behavior (section 4.5.3).

4.5.1 Temporal characteristics



(a) Two-way Modo Weekdays

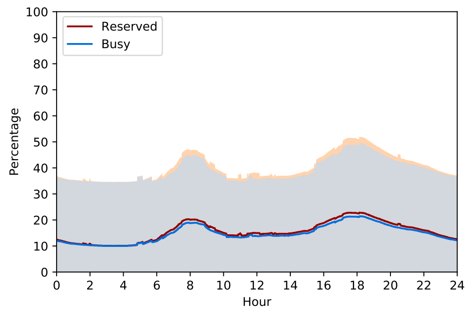


(b) Two-way Modo Weekends

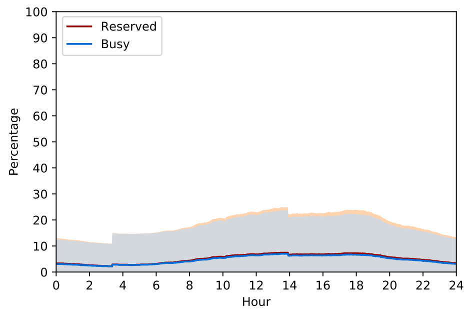
Figure 4.3: Modo Minute-by-minute mean value (plus/minus standard deviation) for the percentage of busy (blue curve) and reserved cars (red curve), for weekdays and weekends

Figures 4.3, 4.4 and 4.5 show the service daily demand pattern. In all plots, the blue and red solid lines refer to a minute-by-minute mean value over the studied period for the percentage of busy and reserved cars, respectively, for each service. The plots show also the standard deviation from the mean as the smoothed gray and orange background areas around the mean. On the left of each pair of figures (figures 4.3a, 4.4a and 4.5a) is presented the demand pattern during working days, while on the (figures 4.3b, 4.4b and 4.5b) is shown the demand for weekends (Saturdays, Sundays, and festivities).

All three services present two peaks of demand during weekdays and only one during the weekends. During weekdays, for Evo and Car2Go, the one-way and free-floating

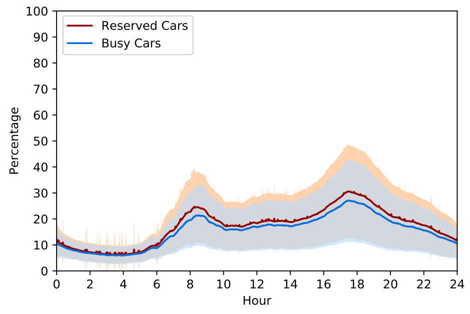


(a) One-way Evo Weekdays

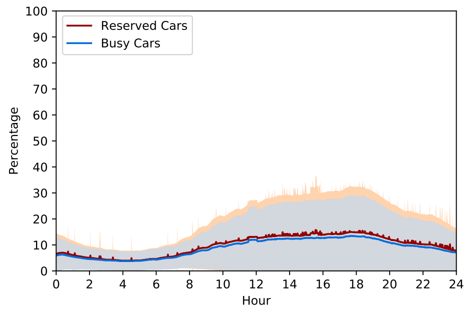


(b) One-way Evo Weekends

Figure 4.4: Evo Minute-by-minute mean value (plus/minus standard deviation) for the percentage of busy (blue curve) and reserved cars (red curve), for weekdays and weekends



(a) Free floating car2Go Weekdays

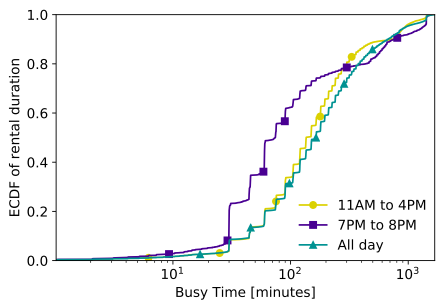


(b) Free floating car2Go Weekends

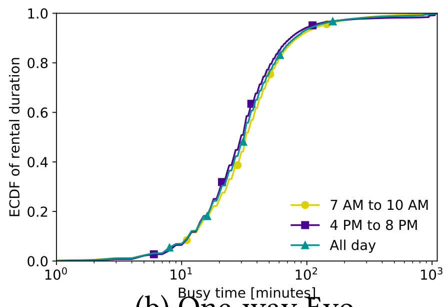
Figure 4.5: car2go Minute-by-minute mean value (plus/minus standard deviation) for the percentage of busy (blue curve) and reserved cars (red curve), for weekdays and weekends

services, the peaks of demand occur about 8 AM and 6 PM, whereas for Modo, the two-way service, these peaks occur around 2 PM and 7 PM. Moreover, note that for Evo and Car2Go, weekdays demand is higher than during weekends. On the other hand, for Modo, we observe just the opposite. Mostly, Modo users are regulars and present weekly/daily/hourly subscription. In this sense, they tend to reserve cars at the same hour, for regular periods, which explains Modo's lower variation. For a given moment, we consider the relative difference between the reserved and busy cars as the cancellations of the system. Modo presents up to 60% of cancellations, while the other two services present no more than 5%.

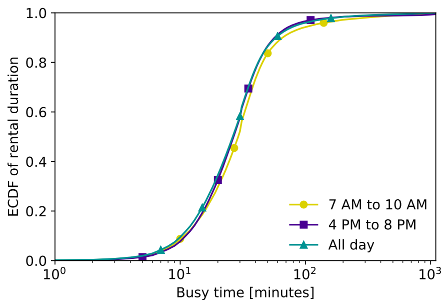
Figure 4.6 presents the Empirical Cumulative Distribution Function (ECDF) of vehicles' busy time, i.e., the rental duration, during load peaks of the day. In this case, we



(a) Two-way Modo



(b) One-way Evo



(c) Free floating Car2Go

Figure 4.6: Cumulative distribution function of vehicle busy time during a weekday.

evaluate the load periods from 7 AM to 10 AM and from 4 PM to 8 PM for free-floating and one-way, from 11 AM to 4 PM and 7 PM to 8 PM for two-way, and also all-day data for the three services.

As for the demand, Evo and Car2Go present similar behavior, which is different from Modo. For Modo, it is possible to observe at least 80% of vehicle rentals present more than 1 hour of occupation, with more than 10% of rentals that last for more than 15 hours. On the other hand, Evo and Car2Go usually present shorter rentals, with no more than 10% of vehicles busy for more than one hour. In sum, the most notable differences between these services occur due to their business model. Indeed, Modo presents a strict policy, where users must pick-up a car and leave it at the same station. However, Modo presents a flexible policy regarding cancellations. The other two services only allow users to cancel the rent of a vehicle up to 30 minutes after its booking.

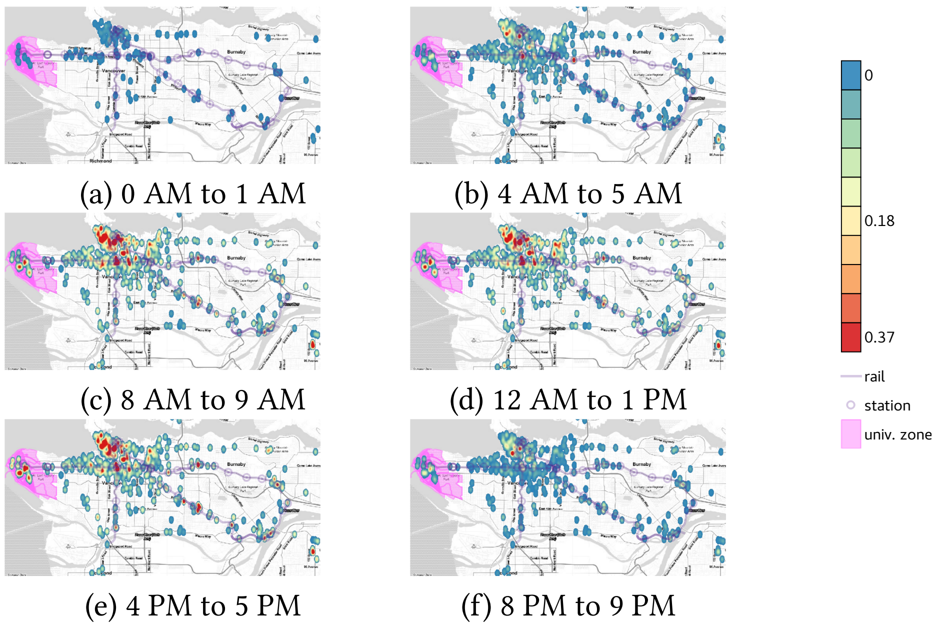


Figure 4.7: Spatial-temporal service demand for two-way service Modo.

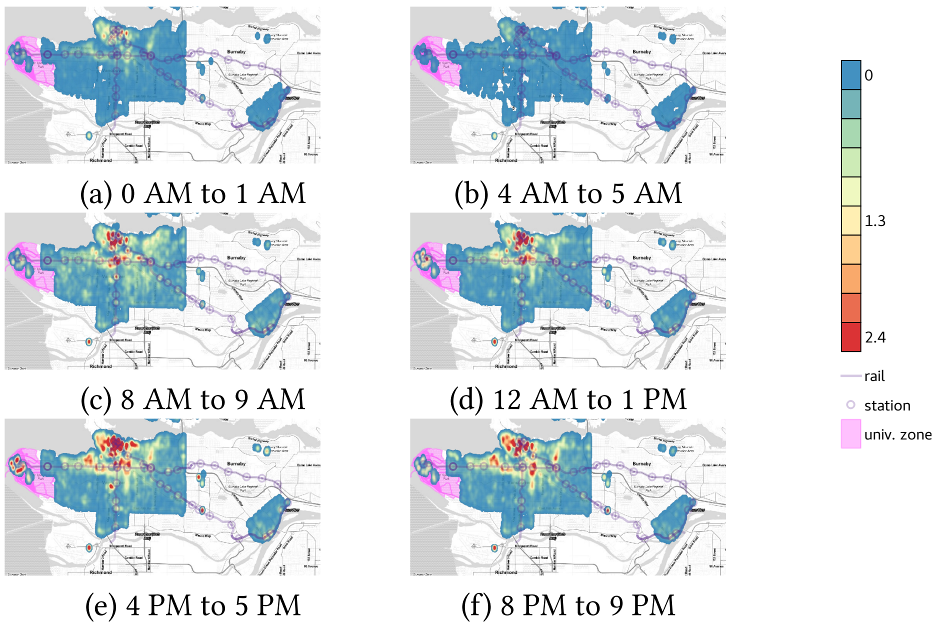
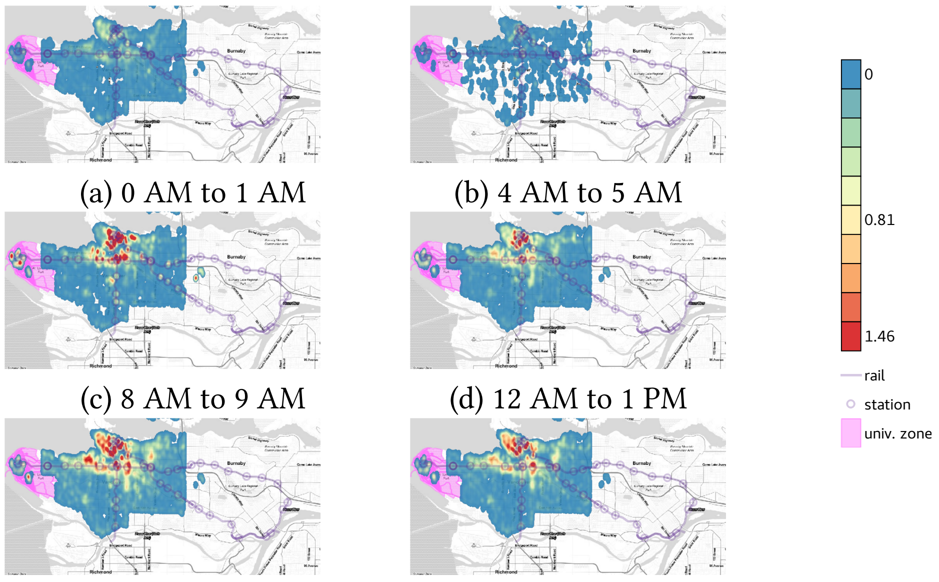


Figure 4.8: Spatial-temporal service demand for one-way service Evo.



(a) 0 AM to 1 AM (b) 4 AM to 5 AM

(c) 8 AM to 9 AM (d) 12 AM to 1 PM

(e) 4 PM to 5 PM (f) 8 PM to 9 PM

Figure 4.9: Spatial-temporal service demand for free-floating service Car2Go.

4.5.2 Spatial-temporal characteristics

Figures 4.7, 4.8 and 4.9 present heat-maps of the hourly[7] mean number of busy vehicles in a given location, considering analyzed period. In the case of Modo, a location refers to a fixed station. In the case of the other two services, the maps present clustered travel records where users pick-up or leave a vehicle. To cluster these points, the algorithm uses a 400 m radius as a reference, forming a region close to a neighborhood. Ranging radius from 100 m to 1000 m, leads to similar results.

First, all three services present a great demand in the downtown area and the university zone. Note that the demand downtown for all three services is low during the night, starts increasing at 4-5 AM, reaches its peak during office working hours, and reduces by the end of the day. In this case, users usually pick-up cars for their daily tasks, as go to work and shopping. During the night, usage increases in the surroundings of the city, the university zone, and neighborhoods with leisure facilities (such as bars). Modo presents a distinct demand pattern. Indeed, Modo has fixed stations located along with the existing public transport system, especially the Expo Line and Millennium Line. For this reason, it is possible to highlight a strong relationship between the existing public transport system and the car-sharing system demand. On the other hand, the other two services are more flexible. Users can rent a car almost anywhere. In this sense, despite the major demand downtown, it is present a widespread demand all over the city.

[7]Due to space constraints, we only show one-hour period every four hour.

Figures 4.10 and 4.11 detail the spatial-temporal demand for Evo and Car2Go by presenting their origin-destination matrix mapped on 31 city areas as defined by the metropolitan city of Vancouver. To enhance the visual effects, we normalized the previous heat-maps values to a scale between 0-1, using the min-max method. Moreover, due to space constraints, this work shows the origin-destination matrix at a specific hour, i.e., at 4 PM. In general, the users tend to start and end a trip at the same location. It appears that during working days, users tend to use a shared car returning it to the same region where they start (likely where they are working or living). However, for both services, a non-negligible probability of spreading services along all city areas is noticeable. Moreover, it is possible to note that some regions serve as hubs. This is more notable for Evo's service. As shown in figure 4.10, the downtown area serves as a hub to start trips to almost all other regions. The opposite (a high tendency to start a trip ending downtown) trend is not present. As a consequence, service may become unbalanced, and, from time to time, service maintenance should relocate vehicles from a region to another to accommodate the daily demand.

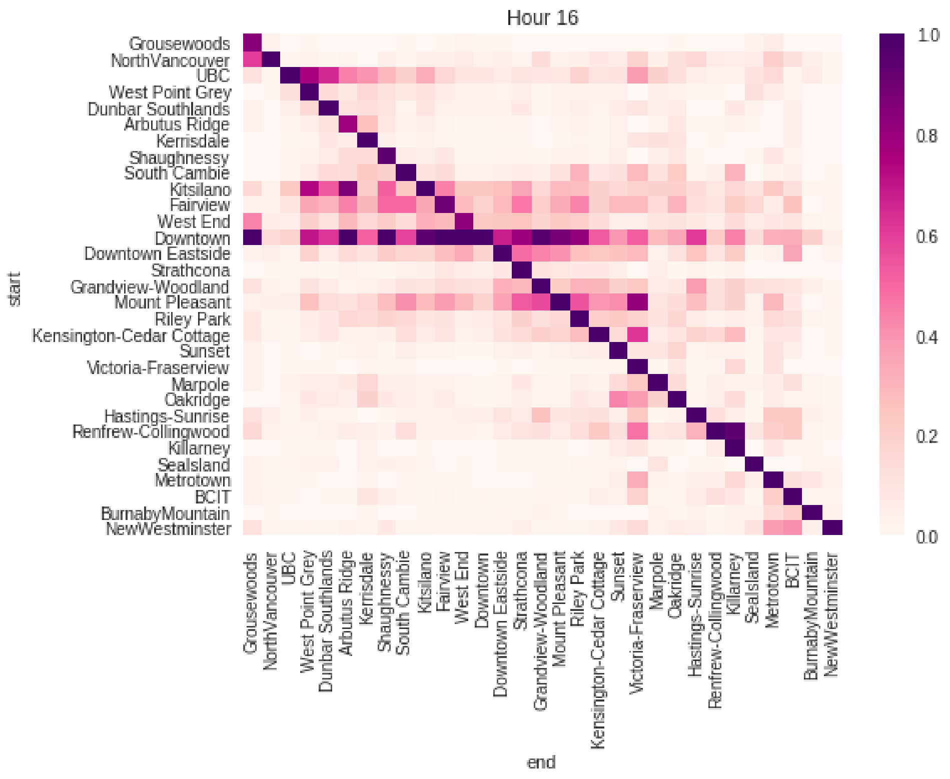


Figure 4.10: Origin-destination matrix for one-way service Evo (from 4 PM to 5 PM).

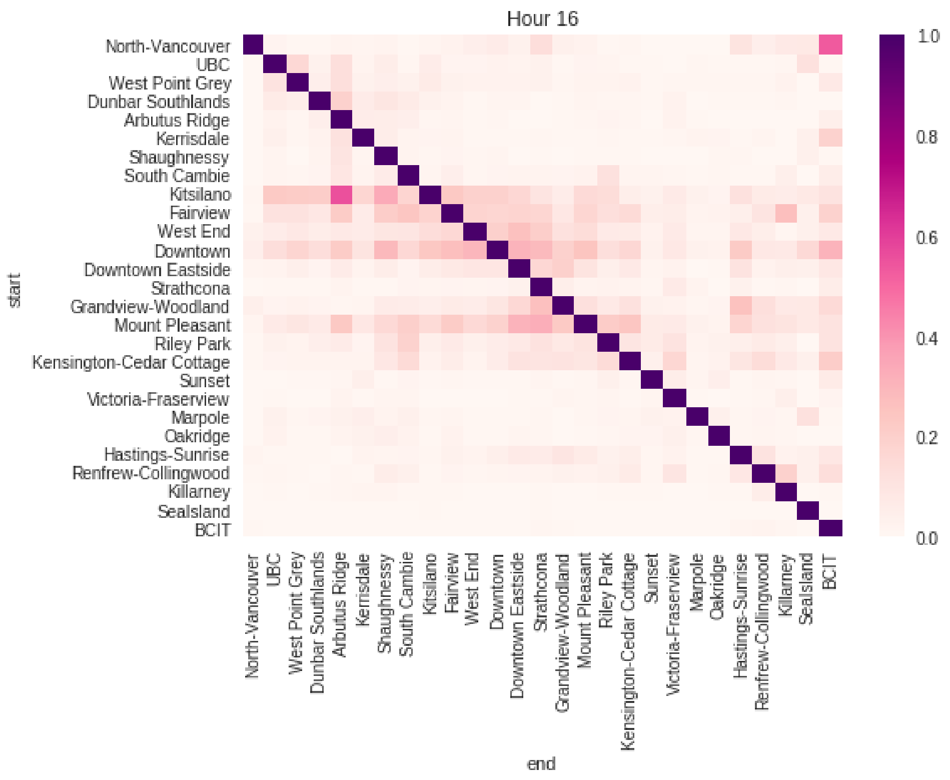


Figure 4.11: Origin-destination matrix for free-floating service Car2Go (from 4 PM to 5 PM).

4.5.3 User behavior characteristics

Vehicle's busy and idle periods direct impacts service revenue. Indeed, the longer the busy period is, and the lower the idle period of a vehicle is, the more profitable the car will be. Therefore, the work characterizes the busy and idle periods of vehicles for all three services. In this analysis, are considered all the vehicles and all the trips lasting less than 90 minutes, which corresponds to more than 99.5% records. For each service, we identified the statistical distribution that best fits the actual data (busy and idle period). For this purpose, we tested more than 40 well-known statistical distributions. More in-depth, for each component of the model, the parameters of the distribution that most closely approximate the data are determined using the Maximum Likelihood Estimation (MLE) method. After defining the parameters of each component of the model, the ten distributions with shorter Kolmogorov-Smirnov distance (continuous distributions) or lower least square error (discrete distributions) concerning the data are chosen. Finally, we chose the top three common distributions to each car-sharing service. These choices are also validated with a visual assessment of the curve fitting.

Figure 4.12 shows the Cumulative Distribution Function of vehicle busy time. Modo, Evo and Car2Go busy time and their best statistical distribution fitting are shown in blue, red and yellow, respectively. For all three services, the Inverse Gamma[8], the Burr[9], and Mielke's Beta-Kappa[10] distributions present a good fitting to the empirical data, with similar MLEs. Table 4.4 summarizes the parameters of the distributions of the busy time for each statistical distribution. Despite all three services present the same statistical distribution fitting, the two-way service (i.e., Modo), presents a clear shift to right on its curve when compared to the other two services, as shown in Figure 4.12. As we previously discussed, the median busy time on Modo is more than one hour longer than the median busy time for the other services. Users in Modo must return cars to the same station they originated travels. As a consequence, they tend to perform longer tasks. On the other hand, with the other two services users tend to do a longer number of shorter travels.

Finally, figure 4.13, presents vehicle idle periods distribution. Power log normal[11], Burr and Mielke's Beta–Kappa distributions best fit the idle data, for all three datasets. Table 4.5 presents the distribution parameters. Again, Modo presents a distinct behavior from the other two services. The longer idle period for Modo vehicles corroborates the previous observations. Indeed, the demand for car-sharing varies over the city during the day. While users in Evo and Car2Go can park anywhere, they contribute to spreading cars over the city. For example, at least 75% of cars in Modo remains idle for periods longer than 2 hours. For the other two services, no more than 20% of vehicles remains idle for the same period.

In sum, the analysis shows that the free-floating and one-way car-sharing systems have similar characteristics. They are mostly used for short/medium period travels, while the two-way system is mostly used for medium to long travels. Moreover, Evo and Car2Go dynamically spread car over the city, turning the car's idle periods shorter. The longer number of shorter travels, associated with the shorter idle periods, may indicate a more profitable service.

4.6 Conclusions

This chapter characterizes three distinct car-sharing systems that operate in Vancouver (Canada) and nearby regions. The study, using data of more than one year of real trips,

[8]Cumulative distribution function (CDF) of the Inverse Gamma distribution: $F(x, a, \beta, \delta) = \frac{1}{\Gamma(a)} \int_{1/((x-\beta)/\delta)}^{\infty} t^{a-1} e^{-t} dt$

[9]Cumulative distribution function (CDF) of the Burr distribution: $F(x, c, d, \beta, \delta) = \left(1 + ((x - \beta)/\delta)^{-c}\right)^{-d}$

[10]Cumulative distribution function (CDF) of the Mielke's Beta-Kappa distribution: $F(x, k, s, \beta, \delta) = \frac{((x-\beta)/\delta)^k}{(1+((x-\beta)/\delta)^s)^{(k*\frac{1}{s})}}$

[11]Cumulative distribution function (CDF) of the Power log normal distribution: $F(x, c, s, \beta, \delta) = 1 - \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-log((x-\beta)/\delta)/s} e^{-t^2/2} dt\right)^c$

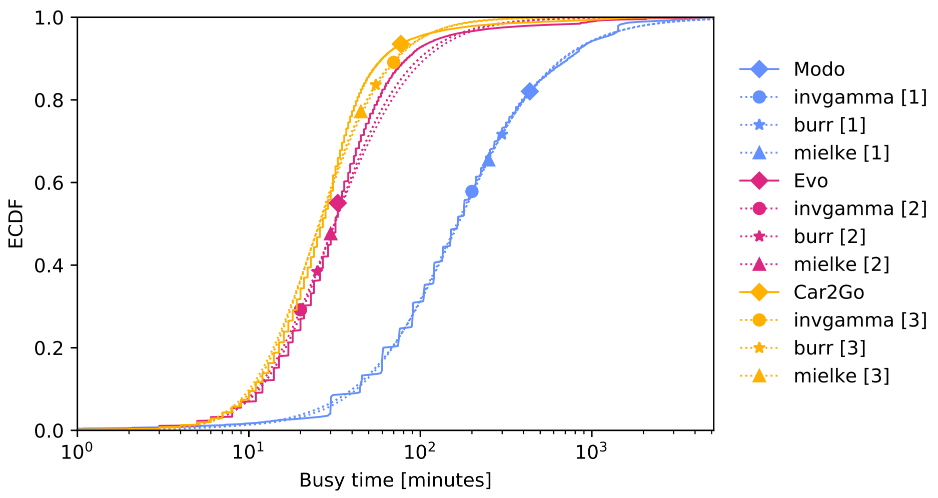


Figure 4.12: Cumulative distribution function of vehicle busy time.

Modo	Inv.Gamma	a = 1.7032, β = -38.5120, δ = 278.8487
	Burr	c = 1.5651, d = 1.0327, β = -1.8893, δ = 163.0525
	Mielke	k = 1.59745, s = 1.5687, β = -1.6713, δ = 164.9877
Evo	Inv.Gamma	a = 2.0674, β = -4.7928, δ = 63.4382
	Burr	c = 1.8332, d = 1.5078, β = -0.1855, δ = 23.5794
	Mielke	k = 2.7305, s = 1.8336, β = -0.1125, δ = 23.7291
Car2Go	Inv.Gamma	a = 2.7688, β = -4.9702, δ = 75.2494
	Burr	c = 2.3869, d = 64.2072, β = -12.5240, δ = 5.7419
	Mielke	k = 37.8163, s = 2.3450, β = -10.9187, δ = 9.6407

Table 4.4: Distributions parameters of the busy time fit curves. The β and δ are key parameters to adjust the location and scale of the distributions.

uncovers patterns of users' habits. We provided a characterization of the different car-sharing services, including spatial-temporal usage. Finally, we highlighted the main differences and the common characteristics of these services.

The analyses point out how in Vancouver in 2017, the one-way and free-floating services were used similarly. They present shorter travels when compared to the two-way service. All three services present peaks of demand during the day. During working days, these peaks occur at around 8 AM and 6 PM, while on weekends, peaks are distributed in the afternoon. The analyzed two-way service presents a considerable number of booking cancellations and a higher vehicle idle time. This indicates a low utilization of the vehicles, likely due to their business model. Indeed, one-way and free-floating services allow users to pick-up a car and leave it anywhere in the city, dynamically satisfying the floating demand. A strong relationship with the public transportation system, as well as with points of interest, such as public universities and

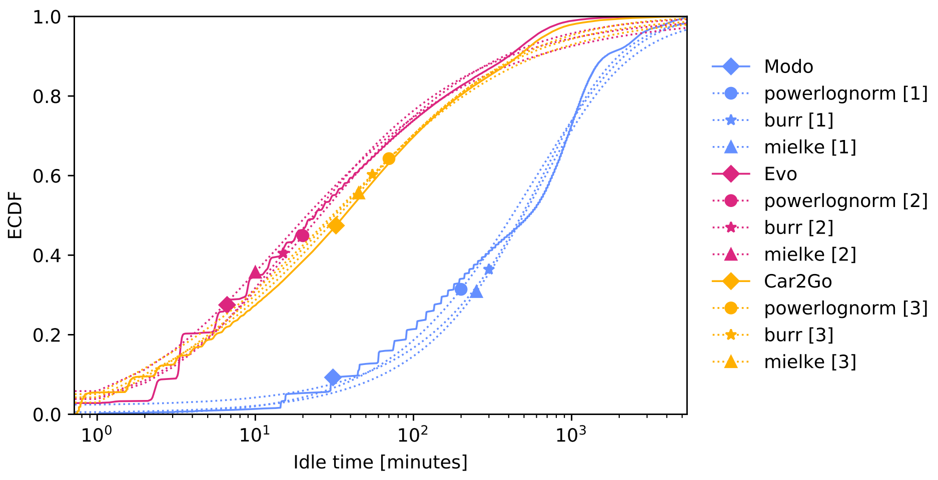


Figure 4.13: Cumulative distribution function of vehicle idle time

Modo	PLogNorm	c = 118.7142, s=3.6088, β=0.7191, δ=3780209.5149
	Burr	c = 1.9865, d = 0.3860, β = -7.7229, δ = 1105.5853
	Mielke	k = 0.8898, s = 1.5390, β = -1.4862, δ = 860.6790
Evo	PLogNorm	c = 0.0723, s = 0.7003, β = -0.6723, δ = 1.8246
	Burr	c = 0.6931, d = 3.7574, β = -0.4881, δ = 2.3713
	Mielke	k = 2.7161, s = 0.5882, β = -0.2800, δ = 0.9725
Car2Go	PLogNorm	c = 4.8747, s = 3.3741, β = 0.7134, δ = 1334.7243
	Burr	c = 0.7714, d = 0.7337, β = 0.7166, δ = 53.9727
	Mielke	k = 0.5743, s = 0.8826, β = 0.7166, δ = 68.1029

Table 4.5: Distributions parameters of the idle time fit curves. The β and δ are keyword parameters to adjust the location and scale of the distributions.

commercial centers, is present.

Chapter 5

Electric Free Floating Car Sharing Mobility Simulator

5.1 Introduction

In the previous chapters, I widely described the dataset I collected and characterized it with quantitative analyses. However, the collected data refers to *only* to internal combustion engine cars. Recalling that the main research question of in this work: *It is possible to design an electric Free Floating Car Sharing System?*, it is now fundamental to introduce a tool to replicate the same FFCS customers' habits in a customizable model of electric FFCS system having both fully electric cars and ad-hoc charging station. In the rest of the thesis I will refer to this particular scenario as **FFCS electrified scenario**.

In this chapter, I describe the core-tool of my research: an event-driven trace-based electric FFCS simulator. In a nutshell, this software is able to (i) extract users pattern from data collected in chapter 2; (ii) replicate the users' patterns in an electrified scenario tunable and parametrizable by the client; (iii) return several performance metrics describing the inputted electric FFCS setup.

In order to give the main idea, the simulator takes as input a real trace composed of an ordered set of rentals. More in details, the simulator creates the event-trace composed by *Event* events. Each one of them carries temporal and spatial coordinates (when the Event triggers), and it can represent a rental start or rental end. In this way, the simulator is able to mirror the exact FFCS trips.

Another input is the operative area composed of adjacent squares zones of 0.025 km^2. Then, each zone may be equipped with a charging station. Therefore, the set of charging stations and their placement is the third input. Complete the input set, the car model, and fleet size.

The simulator detects the closest car to users' requests and moves it from the initial to the final point by consuming the trace. At the same time, it computes the amount of energy needed to afford the trip and properly updates the remaining capacity.

During the simulations, the software computes several metrics in order to measure the proper size of the charging infrastructure, and it impacts on the user discomfort, i.e., in terms of the number of vehicle plugging operations.

Those users' related metrics are heavily influenced by the environmental parameters like the number and the distribution of the charging station. For this reason, the main intuition I followed is to design three placement strategies related to users' driving patterns studying how a given system infrastructure can influence the global system performances.

Moreover, an electric vehicle fleet needs a proper return policy to manage the battery state of charge. Indeed, the long charging time implies a smart car release, especially in zones having a charging station. The simulator takes into account this aspect too and compares different car return strategy.

This chapter is organized as follow: section 5.2 describes the the algorithm behind the simulator, section 5.3 illustrates the charging stations placement, section 5.4 explains how I modelled the provider return policed that customers have to follow, section 5.5 explains the metrics taken in account and measured by the simulator and finally 5.6 concludes the chapter proposing a work resume. More implementation details are reported in appendix A.

5.2 Electric car sharing simulator

The goal is to study different design choices for the electric car-sharing system. For this, I developed a flexible event-based simulator that allows us to compare different algorithms and tune their parameters while collecting metrics of interest. The simulator consumes a trace composed of a subset of rentals collected in chapter 2. In this way, by implementing a module that maps a given car consumption, I can model an electric FFCS provider that exactly replicate customers' temporal and spatial demand.

5.2.1 Simulation model

The simulator replicates the behavior of a fleet of electric cars, which are moving in the city. Each car is characterized by its location and the current status of the battery charge. The simulator takes as input a pre-recorded trace of rentals characterized by the start and end time and initial and final geographic coordinates.

In more details, let is define each trip:

$$i \in \mathcal{I} \tag{5.1}$$

Each trip is characterized by its start and end time, defined as:

$$t_s(i) \tag{5.2}$$

$$t_e(i) \tag{5.3}$$

and origin and destination coordinates defined as:

$$o(i) \tag{5.4}$$

$$d(i) \tag{5.5}$$

For simplicity, I divide the city area into squared zones, of side 500 m. Then, I associate with each position one and only one zone with an unique association defined as follow;

$$O(i) = zone(o(i)) \tag{5.6}$$

$$D(i) = zone(d(i)) \tag{5.7}$$

We assume a charging station cs, composed of k poles, can be placed at the center of a given zone such that

$$z \in \mathscr{Z} \tag{5.8}$$

so either:

$$cs(z) = \begin{cases} 1, & \text{if } cs \text{ is present} \\ 0, & \text{if } n \text{ otherwise} \end{cases} \tag{5.9}$$

The total number of zones equipped with charging stations is defined as follow:

$$N = \sum_{z \in \mathscr{Z}} cs(z) \tag{5.10}$$

with $K = N \cdot k$ the total number of poles.

Additionally, it is present a set $\mathscr{A}$ of cars, with its cardinality $|\mathscr{A}|$ obtained by the trace. More in details, let it define each car

$$a \in \mathscr{A} \tag{5.11}$$

at time t is characterized by its position:

$$p(a, t) \tag{5.12}$$

and its zone:

$$P(a, t) = zone(p(a, t)) \tag{5.13}$$

and the residual battery capacity:

$$c(a, t) \in [0, C] \tag{5.14}$$

with C being the maximum nominal capacity.

Generally speaking, the simulator processes each rental event i in temporal order. When a *rental-start* event i is processed at time $t = t_s(i)$, the simulator chooses randomly one of the most charged available car in the closest zones to the initial position zone $O(i)$. In formulas, we get a car $\bar{a} \in \mathscr{A}$ such that:

$$c(\bar{a},t) \geq c(\hat{a},t)\ \forall \hat{a} \in \arg\min_{a\in A} dist(O(i), P(a,t)). \tag{5.15}$$

The simulator mimics the normal behavior of FFCS customers that use their smartphone to rent the closest car from their position and are worried about vehicle range [47]. Notice that this behavior is independent of whether the car is at a pole being charged or not. Then, the simulator schedules the event *rental-end*, and it makes the car unrentable. When the rental ends fires, all the statistics about the rented car are updated (like battery consumption and new destination). Obviously, the simulator is able to manage all the events, like battery depletion or unavailable cars nearby the rental starts.

In the output, the simulator produces several statistics about system usage and user-related discomfort metrics related to the electric vehicle plugging procedures.

5.2.2 Modelling of rental event

When a *rental-start* event i is processed at time $t = t_s(i)$, and the simulator looks for a car in the initial position zone $O(i)$. If one or more cars are present, it selects (one among) the most charged car, i.e, get the car $a \in \mathscr{A}$ such that

$$P(a,t) = O(i) \wedge c(a,t) \geq c(a',t)\ \forall a' \mid P(a',t) = O(i), \tag{5.16}$$

independently whether the car is at a pole being charged or not.[1]

If any car is available, the simulator selects the closest zone to $O(i)$ containing an available car, mimicking the normal behavior of FFCS customers that use their smartphone to rent the closest car from their position. If any vehicle is present in the 8 eight neighboring zones, the rental is marked as *infeasible*. A *rental-end* event is then scheduled using the trace final time $t_e(i)$ and location $d(i)$.

When car a rental-end event is processed at time $t_e(i)$, the simulator makes as available the car in the real position $p(a, t_e(i))$. The arrival zones might correspond to the one present in the *rental-end* event, or it might be necessary to manage a slightly user's rerouting due to vehicle plugging procedures. The policies which regulate when and how plug the car are described in section 5.4. Once the car is released, the simulator updates the battery State of Charge (SoC) by consuming an amount of energy proportional to the real trip distance:

$$c(a,t_e(i)) = \max\left(c(a,t_s(i)) - Energy(p(a,t_s(i)), p(a,t_e(i))), 0\right) \tag{5.17}$$

with $Energy(\cdot)$ that models the energy consumed to go from the car origin $p(a,t_s(i))$ to the car destination $p(a,t_e(i))$. In case $c(a,t_e(i)) = 0$, the trip i is declared *infeasible*. The discharged car a still performs further trips, all marked as infeasible, until it reaches a charging station.

[1]We choose this policy because people are worried about vehicle range [47].

5.3 Meta-Heuristic Charging Stations Placement

In this section, I explain the charging station placement algorithm. The output of this algorithm is one of the most relevant environmental variables that will be deeply studied and analyzed in chapters 6 and 7. The main idea behind this algorithm to rank each zone (defined in section 5.2) according to the users' traveling patterns and, then equip zones having the highest values.

5.3.1 Problem formalization

Given a number of charging station N, the first objective is to place them in the city area to let all rentals feasible, i.e., to find a charging stations placement so that

$$c(a, t_e(i)) > 0 \ \forall a \in \mathscr{A}, \forall i \in \mathscr{I} \tag{5.18}$$

Since I do not make any assumption on the set of trips $\mathscr{I}$, I cannot know a-priori if a solution exists and provide a general analytical solution. Moreover, the number of candidate solutions increases as the binomial coefficient $\binom{|\mathscr{Z}|}{N}$, making it ineffective to numerically compute all possibilities. Instead, I will provide a class of greedy algorithms and analyze the performance in our specific cases of $\mathscr{I}$. In details, each zone $z \in \mathscr{Z}$ is assigned a likelihood $l_z \geq 0$. We then solve the problem of finding the subset of N zones that maximizes the total likelihood. In formulas,

$$\max \sum_{z \in \mathscr{Z}} cs(z) l_z \tag{5.19}$$

subject to:

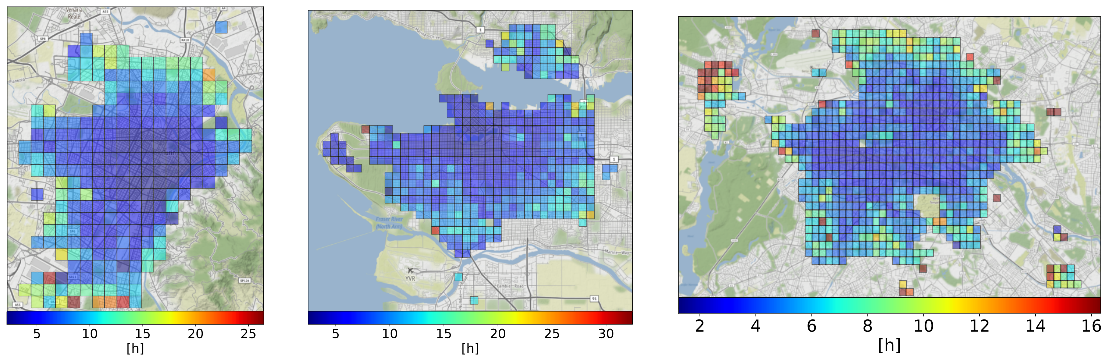
$$\sum_{z \in \mathscr{Z}} cs(z) = N \tag{5.20}$$

$$cs(z) \in \{0,1\}, \forall z \in \mathscr{Z} \tag{5.21}$$

The above optimization problem can be solved by greedily choosing the top N zones, ordered in decreasing likelihood. We compare the performance of different placement algorithms based on a different definition of the likelihood.

- *Random placement*: l_z is an independent and identical distributed random uniform variable so that charging stations result placed at random;
- *Average parking time*: l_z is the average parking duration in z as recorded in the trace;
- *Total number of parkings*: l_z is the total number of parking events recorded in z in the trace;
- *Total parking time*: l_z is the total parking time accumulated in z by all cars recorded in the trace. In each zone, it is the product of the two previous metrics.

Those heuristics are driven by the intuition that placing charging stations in those zones where cars are parked for a long time (average parking time) or frequently parked (total number of parking) could improve system performance.



(a) Turin (b) Vancouver (c) Berlin

Figure 5.1: Distribution of average parking time in Turin, Vancouver and Berlin

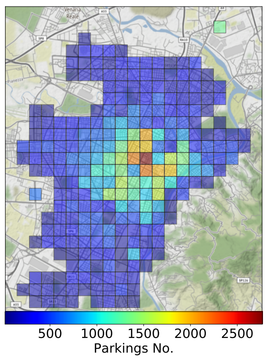
I order to show the differences between the likelihoods l_z criteria, figures 5.1 where l_z is depicted for Turin (5.1a), Vancouver (5.1b) and Berlin (5.1c). The first two cities were deeply characterized in chapters 3 and 4, while Berlin, as I will show, presents some interesting spatial distribution. In all the figures, the more the zone is red, the higher is l_z. It means that the *redest* zones will be the first to host a charging station.

In first approach, it is possible to see how, in all figures, the heuristic *Average parking time* is mainly spread in city peripheries. It means that cars spend a lot of time parked far from the city center. This peculiarity can be imputed to commuting patterns: as figure 3.2 points out, two peaks are present in the users' demand. In particular, the evening peak catches the back-home commuting which, usually is directed to the high-density residential area located in the periphery. This, joint with the low business-days night demand, leads to users to leave cars parked in those areas all night long.

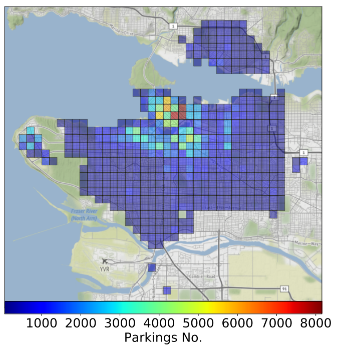
Figure 5.2 depicts the number of parking in each zone, for Turin (5.2a), Vancouver (5.2b) and Berlin (5.2c). Reminding that more parking means higher zone attractiveness, it is possible to notice how the zones with the highest number of parking concentrations are delimited in particular areas. For example, figure 5.2a shows how most frequented areas are downtown in correspondence of the two main train stations and the airport. A similar pattern can be spot in figure 5.2b. Contrary, Berlin presents at least three attractive areas. This is mainly due to the biggest operative area and, probably, to the differentiation of business areas.

For completeness, I report in figure 5.3 the *Total parking time* likelihood. It appears to smooth the behavior of the previous two metrics.

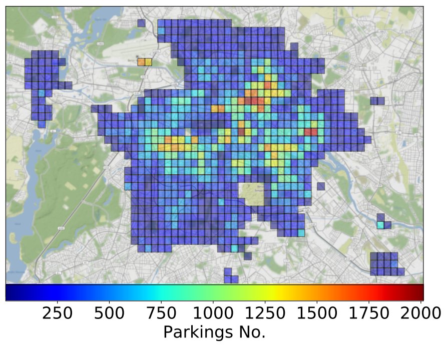
This brief catheterization shows how different cities can have different spatial characterization and, thus, different charging station placements. However, those characterizations will be deepened in chapter 6.



(a) Turin

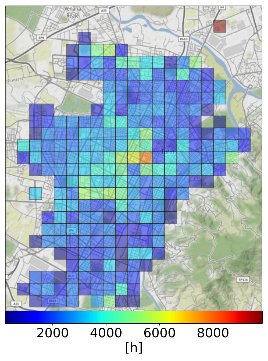


(b) Vancouver

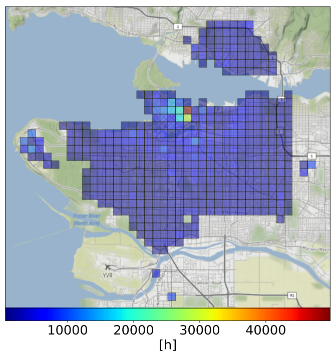


(c) Berlin

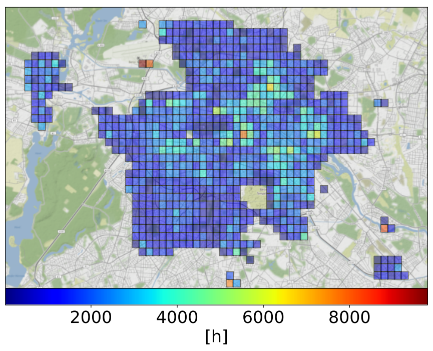
Figure 5.2: Distribution of number of parking in Turin, Vancouver and Berlin



(a) Turin



(b) Vancouver



(c) Berlin

Figure 5.3: Distribution of total parking time in Turin, Vancouver and Berlin

5.4 Car return policies

One of the most challenging points of electric FFCS is to deal with the discomfort derived by plugin operations. This operation is more time-consuming compared to the normal filling up procedure of internal combustion engine cars. Therefore, the providers have to deal with users' selfishness and trying to stimulate their willingness.

When returning the car, the customer may connect the car to a pole in a station, hence charging the car battery and possibly deviating the real destination from the desired one. I modeled the following policies:

- *Free Floating*: the customer must connect the car to a charging pole if and only if it is available in the desired final zone $D(i)$;
- *Forced*: cars must be connected to a pole when the percentage of battery charge at the end of the rental i would go below a certain threshold π as the following

expression defines:

$$(c(a, t_s(i)) - Energy(p(a, t_s(i)), d(i))) \cdot 100/C \leq \pi \tag{5.22}$$

This implies the customer can be *rerouted* to the closest zone to the desired one $d(i)$, if no free pole exists in the zone;

- *Hybrid*: the customers follow the forced policy; they may also choose to connect to a charging pole available in the desired ending zone $D(i)$ with probability $w \in [0,1]$;

The *Free Floating* policy never obliges the customer to bring the car far from the desired ending location, even in case the battery is close to exhaustion. It benchmarks the other policies in order to understand until when the users might rent a car without any restriction/ *Forced* mandates to connect cars to a charge station only when energy runs low, thus trying to protect from battery exhaustion. *Hybrid* introduces the level of customers willing to collaborate, named with w. $w = 0$ is equivalent to the Forced policy, while $w = 1$ adds to the Forced policy the Free Floating policy, thus always connecting the car to a charging pole if available in their final position zone. The users' willingness should be w should be intended as the probability that a user can collaborate with the provider, dropping the car in a charging station. The w variability can be justified like provider incentive bonus like car2go free minutes after a car filling up.

5.5 Key Performance Indicators and Simulation Scenario

In this section, I describe which are the simulation outputs and the scenario with which I performed the analyses. In particular, I focused the attention on minimum requirements to system sustainability and measuring users' discomfort.

5.5.1 Performance metrics and parameters

The simulator measures metrics that are key to assess in the quality of experience for the customers:

- *Infeasible trip*: measures if a trip i performed by a car a ends with a completely discharged battery, i.e., when $c(a, t_e(i)) = 0$;
- *Charge event*: indicates a trip i that ends with putting in charge the car, implying the burden to drive to the pole position, and plug the car;
- *Reroute event*: a trip i where the customer is rerouted to a zone different from the desired destination because forced to charge the car a, i.e., $P(a, t_e(i)) \neq D(i)$;
- *Walk distance*: distance between the desired final location $d(i)$ and the actual final position $p(a, t_{end}(i))$.

The number of infeasible trips is critical, and the system shall be engineered so that they never happen. Other performance metrics shall be minimized. In addition to the above metrics, the simulator collects statistics about car battery charge level $c(a, t)$, and the fraction of time a battery stays undercharge.

5.5.2 Simulation scenario

I use this simulator to study the system feasibility of an electric FFCS at the varying of the number of zones that are equipped with charging stations N, and the number of poles k of each charging station.

I consider in each city a fleet that has a number of cars equal to the one observed in the trace. Electric cars have the same nominal characteristics as the Smart ForTwo Electric Drive, i.e., 17.6 kWh battery, for 135 km of range, with a discharge curve $Energy()$ that is proportional to the traveled distance (12.9 $kWh/100\,km$). [2] Charging stations have $k = 4$ low power (2 kW) poles each. These are cheap to install and a good compromise between costs, power requested, and occupied road section. We model a simple linear charge profile (complete charge in 8 hours and 50 minutes in our case). At last, the initial car position, only affecting the simulation transient, is chosen randomly.

The simulator, written in Python, takes less than 5 seconds to complete a single simulation for a given city and parameter set. Due to a large number of simulations, we run them in parallel. Each simulation produces 100 MB of detailed logs that we process on a Big Data cluster of 30 nodes using PySpark.

5.6 Conclusions

In this chapter, I described a FFCS electric mobility simulator I developed. Starting from the data collected with the software described in chapter 2 I created a trace of rental events, describing the system allocated users' demand.

More in detail, the simulator allocates a set of cars, characterized by battery capacity and power consumption per kilometer. Then consumes the rental trace, marking the car unavailable after a *rental-start* event and updating the final battery state of charge when a *rental-end* event is processed. Moreover, the simulator is in charge of placing the charging station according to three heuristics: random, preferring zones having a greater parking time, and zones having a higher number of parking events. Finally, it takes into account the different policies with which the users have to return the car.

When the trace is consumed, this simulator computes several key performance indicators measuring the proper system infrastructure allocation and users' discomfort to deal with an electric vehicle.

[2]https://www.smart.com/uk/en/index/smart-electric-drive.html

Chapter 6

A Data Driven Approach for Electric FFCS System Design

This chapter refers mostly two works: *"Free floating electric car sharing in smart cities: Data driven system dimensioning"* published in the IEEE International Conference on Smart Computing (SMARTCOMP) in July 2018 [48] and *"Free Floating Electric Car Sharing: A Data Driven Approach for System Design"* published in the IEEE Transactions on Intelligent Transportation Systems journal in August 2019 [43]. I am main the main contributor of this paper while other authors supervised the entire publishing process.

6.1 Introduction

Nowadays, mobility is a very important challenge for our society, with strong implications on pollution in large cities where more eco-sustainable solutions are positively seen as a means to improve the current situation. Along with the usage of public transport, sharing mobility such as bike-sharing, carpooling, and car-sharing can help to address this problem. In this chapter, I focus on the design of an electric car-sharing system, where customers rent a car for moving within the city limits for short periods of time. I focus on the so-called Free Floating Car Sharing (FFCS) system where customers are free to pick and return the car wherever they like, inside a geofenced area. Electric car-sharing systems need an infrastructure of recharging stations, whose design requires ingenuity [49, 50, 51].

Data is fundamental to answer these design questions. In this work, I base the study on the availability of millions of actual rentals I collected from currently in use FFCS systems as reported in chapter 2. In this chapter, I consider Turin and Vancouver (characterized in chapters 3 and 4) plus Berlin(Germany) and Milan(Italy), which in the dataset are the cities having the highest number of rentals. The data naturally factors the non-stationarity of FFCS systems, including millions of actual rentals. Armed with this, I study and compare the performance of a hypothetical equivalent car-sharing system based on electric vehicles. While in the past some works have proposed solutions for the design of electric FFCS [52, 53] and for a smart placement of charging stations, this work is among the first to take a data-driven approach for the design of electric car FFCS systems [54, 51, 49, 50].

First, I characterize how customers actually use FFCS transport means in different cities and countries. Results show a similar usage with a high utilization during a commuting time and very different spatial distributions. Rental duration and driving distance are quite short (less than 20-30 minutes, for less than 5 km in median). More interestingly, it is possible to observe peripheral zones where cars are left parked for a long time and busy areas where instead the parking duration is much short and dynamic.

Armed with these facts, I compare charging station placement policies introduced in section 5.3 that exploit the knowledge of typical parking zones and duration. I first assume a pure Free Floating system, where customers return the car in a charging station only if present at their actual destination. Results show that placing the charging stations in those areas where cars stay parked for a long time performs badly. Instead, placing charging stations in those areas where cars are frequently parked and rented, e.g., near train stations and working areas, guarantees much better performance. This is consistent in all cities.

Next, I study different return policies introduced in section 5.4, where customers are asked to return the car to a charging station in case the battery level decreases below a minimum threshold. This collaborative policy reduces the cost of the charging infrastructure by a factor of 2 or more with respect to pure opportunistic free-floating solutions. Equipping just 8% of charging zones with 4 poles of 2 kW would guarantee an electric car FFCS equivalent to the one currently in use. This with minimal impact on customer satisfaction, measured by the number of times customers are forced to drive to a charging station and the distance they have to walk back to the desired destination.

At last, I compare system design alternatives to check whether it is better to place a lot of charging poles in very few areas, or rather to spread a lot of charging stations with few poles in many areas. Results demonstrate that both extreme solutions perform badly, with the best performance when installing charging stations with 5 to 20 poles in popular areas. This has benefits also on the power grid used to supply power to the recharging areas.

After quickly discussing related work in section 6.2, I present and characterize data in section 6.3, section 6.4 discusses the impact of charging stations placement policies, while section 6.5 compares return policies. Section 6.6 presents the impact of concentrating or spreading charging stations in the city. Finally section 6.7 concludes the chapter.

6.2 Related work

The diffusion of the free-floating approach to car sharing led to increasing attention by many researchers, with analyses of these systems and their extension to electric vehicles. The studies performed in 2011 by Finkorn and Müller [27, 55] are the first attempts to analyze the benefits of FFCS for the population. Their results on customers' behavior, like traveled distances, are similar to what I exposed in chapter 3. Later works [56, 57, 22] also collected data and analyzed the mobility patterns of customers and differences among cities.

The introduction of electric vehicles for private and public transportation brought the problem of placing the electric charging stations. Authors in [54] show the benefits of placing charging stations with different power according to the customer parking duration. Few data-driven studies address the charging station placement, either by respectively minimizing the cost of installation, power loss, and maintenance [49, 51], or by minimizing the customers' walked distances necessary to reach a charging pole[50].

After a survey among FFCS customers in Ulm (Germany), authors of [52] investigate the positive influence and feasibility of electric FFCS systems. Lastly, authors of [53] study the relocation of electric cars in FFCS since few charging stations may be blocked by completely charged vehicles.

Previously, in [48], I performed several analyses about how to design an electrical FFCS in the city of Turin. In [58] I introduced the first optimization of electric charging stations placement. In this work, I extend both works by considering four cities as a case study and studying a new set of return policies to observe the impact of the willingness of customers to contribute to the system's sustainability. I further extend this work by discussing the benefits of using charging hubs.

In this work, I should be among the first to validate a data driven approach for dimensioning an electric FFCS system by analyzing and optimizing different metrics impacting customer experience in different cities worldwide located.

6.3 Data collection and characterization

Here I characterize system usage in all the case of study cities, focusing on those metrics that are instrumental for the design of FFCS systems based on electric vehicles and highlighting the non-stationary patterns.

6.3.1 Temporal characterization

I collected the data with UMAP, described in chapter 2. Recalling that the basic dataset entity is the *bookings*, formalized in definition 2. Not all bookings correspond to an actual real user's trip. Indeed it is possible to observe the booking dataset is composed by *reservations* and *rentals*, defined respectively in definition 3 and definition 4. This ambiguity is due to the high degree of freedom of an FFCS system:

- a customer can book a car, and cancel the booking later on;
- the data collection may suffer from outages, so that some snapshots may miss some available cars;
- cars may go in maintenance so that they disappear and never come back (or return after a long time)

To tackle this problem, I developed data cleaning and filtering rules to extract actual *rentals* from bookings date-set. A rental is identified when:

- it lasts at least 3 minutes;
- the ending position is at least 700 m far from the starting position, with both positions inside the city operative area;[1];
- its duration is smaller than 1 h. These thresholds have been selected by domain knowledge of the data – see 3 for more details.

Bookings that do not correspond to rentals are then merged with parkings events (since the car did not move).

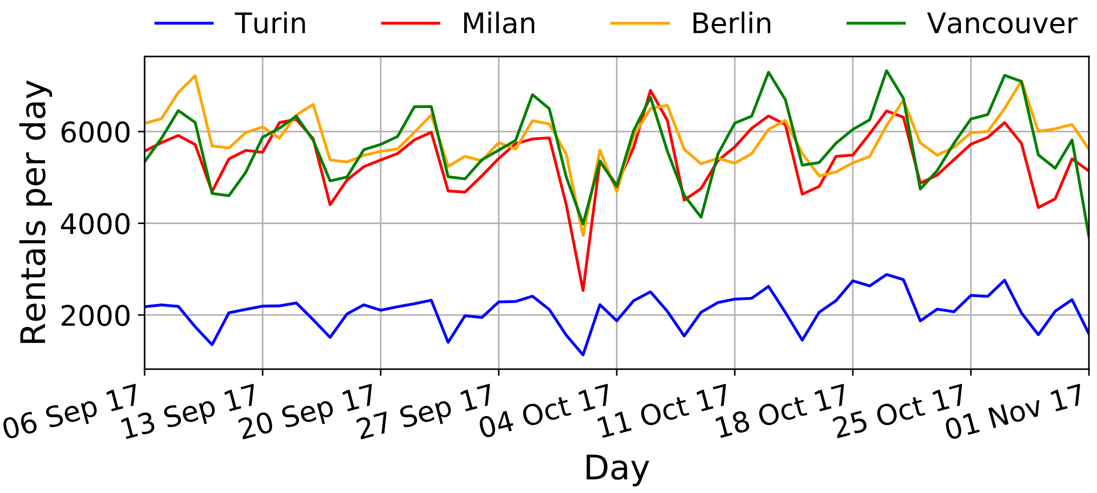
In this work, I focus the analysis from September to November 2017 in four cities: Turin (Italy), Milan (Italy), Berlin (Germany), Vancouver (Canada). Overall, the case study dataset has more than 1 million rental events that describe the typical usage patterns of FFCS customers.

To give the intuition of the system, I first provide a characterization of actual usage patterns by current FFCS customers in each city. I focus first on temporal characteristics. Figure 6.1 reports the rental trend. More in detail, figure 6.1a shows the number of recorded rentals for each day in the considered period. Usage similarity is striking, with Milan, Berlin, and Vancouver that have more rentals per day than Turin. This intense usage justifies the difference in fleet size between the cities, with these three having at least twice as many cars with respect to Turin (see table. 6.1). These first results highlight the importance of extending the users' pattern analyses of the same provider in different cities. A second interesting aspect is the presence of a weekly pattern: in correspondence with the weekends, the number of rentals drops by about 30%. This is justified by the fact that during the working days' cars are used for commuting. Moreover, non-stationary events due to holidays or strikes are visible, e.g., October 6th in Milan due to a public transport strike.[2]

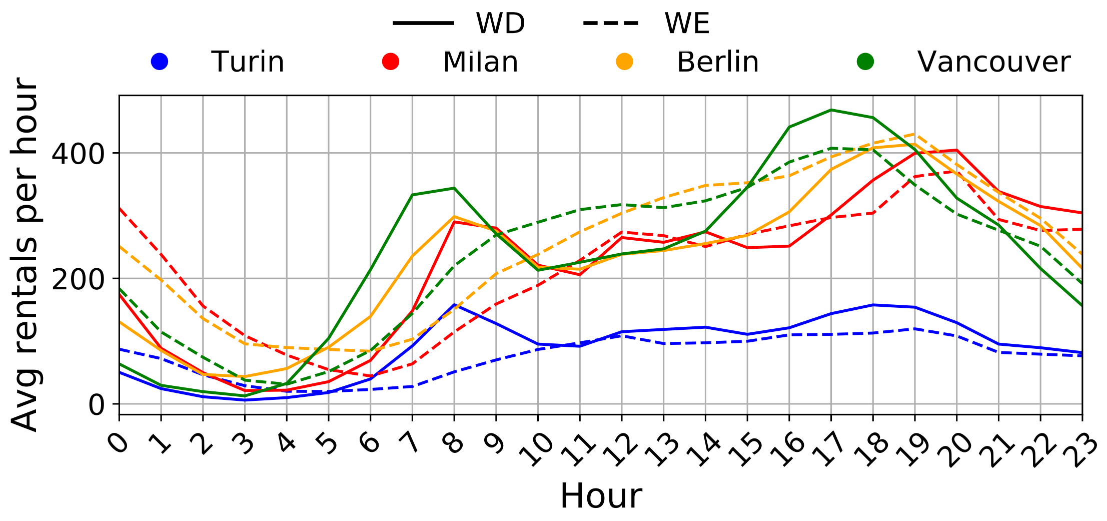
To deeply analyze customers' habits, I detail the average number of rentals per hour in figure 6.1b, separately per working-days (WD, solid line) and per weekends (WE,

[1]This to account for possible errors in the GPS fixing, and to remove rentals started and ended in different cities.

[2]The sudden fall around the October 2th is due to a system outage that caused a lack of data.



(a) Number of rentals per day.



(b) Average number of rentals per hour.

Figure 6.1: Rental trends in different cities.

dashed line). Each curve reports the number of rentals for each hour, considering the average number over the same hour in the dataset. Firstly, notice the usage peaks during commuting times. These happen at different times for different cities, e.g., 8 am for Turin, Milan and Berlin vs 7am for Vancouver, following local commuting habits. Secondly, notice how the evening and night usage tend to be larger during weekends than working days. This reflects the different usage patterns at night when cars are used to reach areas dense with pubs and nightlife. At last, observe the different patterns again

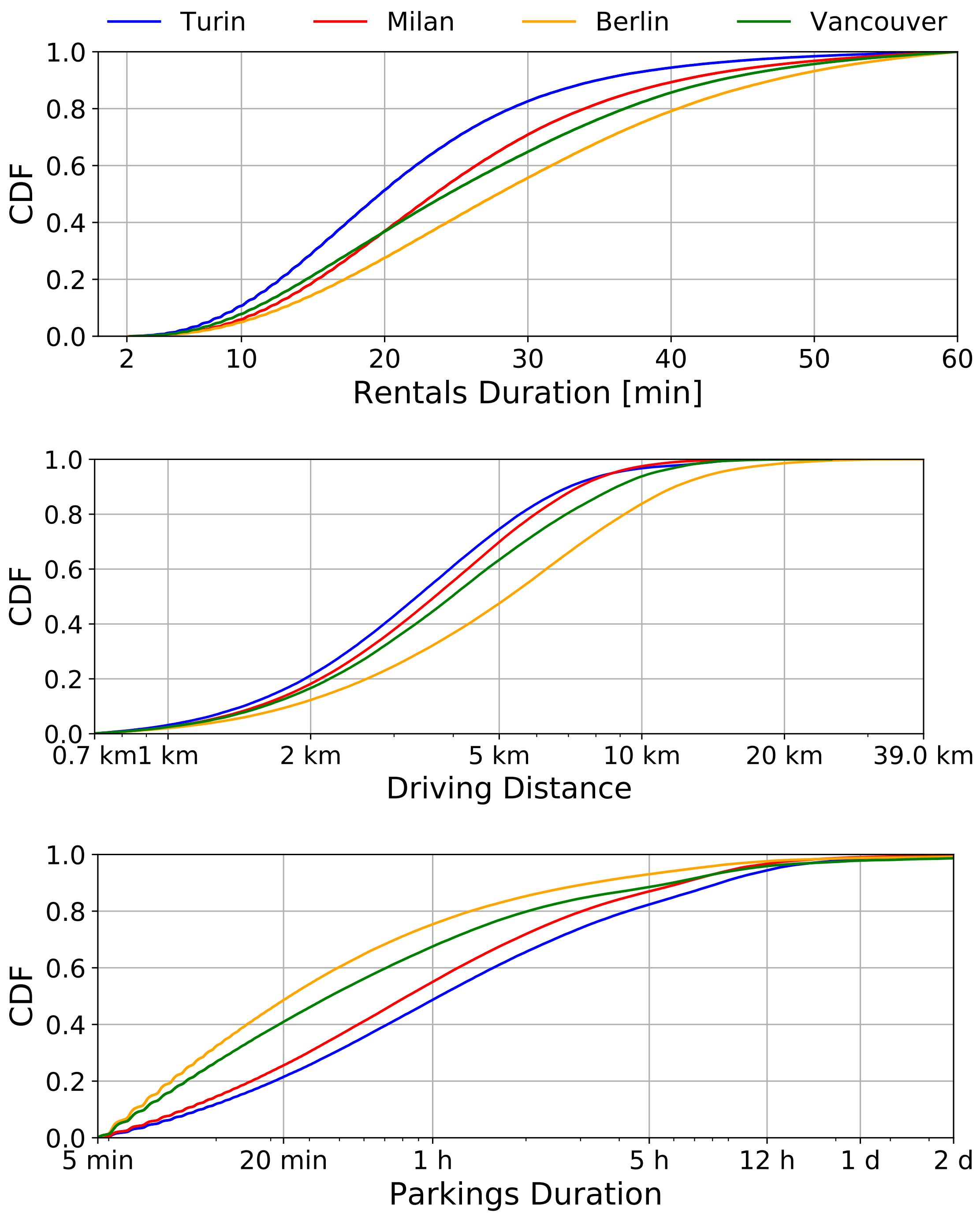
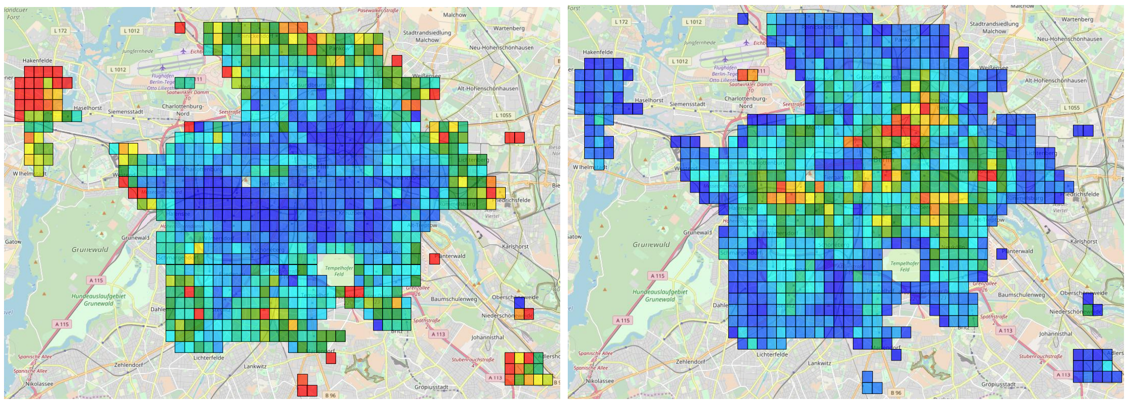
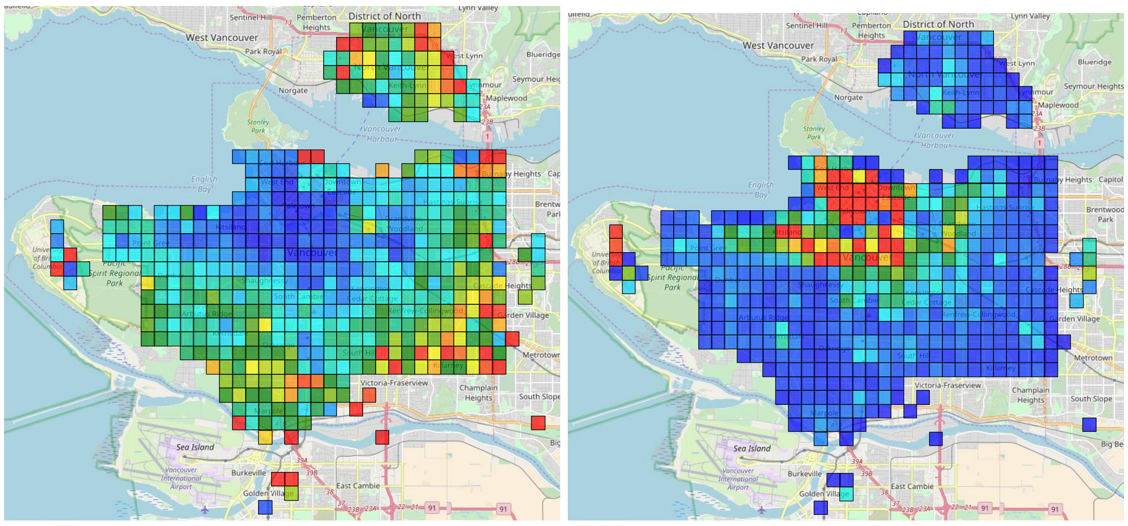


Figure 6.2: Car Sharing usage habit characterization.

in different cities. For instance, the average number of rentals in Vancouver and Berlin during weekend mornings is higher than during working days. This does not happen in



(a) Berlin: average parking time (left) and total number of parkings (right).



(b) Vancouver: average parking time (left) and total number of parkings (right).

Figure 6.3: Heat map of average parking time and total number of parking. The warmer the color, the higher the value.

Italian cities. The charging station placement design must thus weight these different needs and non-stationary patterns.

Given the goals of deriving guidelines for charging station placement policies, I focus now on the characterization of three important metrics: (i) for how long customers rent a car, (ii) how far they drive, (iii) how long cars usually stay parked. The former two metrics guide the battery discharging properties, while the latter metric is fundamental to understand battery charging opportunities. Given the dataset does not have any information on car position during a rental, I compute the travel distance by assuming the customer went directly from the origin to the destination. This is indeed compatible with the duration of rentals (see below). I used Google Map service to compute a correcting factor to be applied on the euclidean distance as described in chapter 3.

Figure 6.2 reports the Cumulative Distribution Function (CDF) of the rentals duration (top), driving distance (middle) and parking duration (bottom). The size of the city has a clear impact, with Turin that has the shortest trips, and Berlin the longest. The rental duration is, in general, very short, leading to the intuition that drivers tend to minimize the rental time (and cost). Traveled distance is fundamental to understand the battery consumption: The maximum driving distance sets the minimum battery charge to sustain that trip. Looking at the middle plot in digure 6.2, I observe that in Berlin, the longest trips are twice as long as the longest trips in other cities. Therefore, the same battery constraints would not fit for all cities.

Overall, the limited traveled distance and rental duration suggest people use the car just for the time strictly needed to reach their destinations. Trips are limited by the service area and are thus typically within 15 km (25 km for Berlin).

At last, a bottom plot of figure 6.2 details the duration of the parking periods. Interestingly, 50% of parking lasts less than 22 minutes in Berlin, testifying to a very high system utilization. In Turin, the median grows to 42 minutes, still showing that most cars are parked for a short time. Yet, the long tail of the CDF (note the log scale on the x-axis) suggests that there are a sizeable fraction of parkings that last for 5 or more hours. Cars parked in the periphery, typically at night, where the demand is lower, belongs to this fraction.

Table 6.1: Main characteristics of data.

City	Rental	Fleet Size	Rental Time [min]		Rental Dist. [km]		Parking Time [h]		Zones
			Avg	Med	Avg	Med	Avg	Med	
Torino	125k	377	21	20	3.96	3.36	3:17	0:42	261
Milano	320k	739	25	24	4.15	3.66	2:21	0:24	549
Berlino	342k	900	29	28	6.22	5.24	2:23	0:22	833
Vancouver	317k	941	26	24	4.70	3.98	2:54	0:22	532

Takeaway: car sharing usage is time heterogeneous and non-stationary. However, many patterns can be identified. FFCS customers tend to use the system mostly during the commuting time and for short trips.

6.3.2 Spatial characterization

The choice of charging station placement has to balance two main factors: place them where cars are (i) frequently parked – so to maximize the opportunity for charging; and where cars (ii) stay parked for a long time – so to maximize the charging time. Knowing the zones within the city where cars are left parked is fundamental. For this, I divide the operative area described in section 5.3

Figure 6.3 shows the above metrics for Berlin and Vancouver using a heat map – with blue and red corresponding to the minimum and maximum values, respectively. Focus on Berlin first - figure 6.3a. The left plot shows the average parking time. Results clearly show that cars stay parked for a very short time in busy downtown areas, while cars stay parked for a longer time in the periphery (corresponding to the head and tail of the CDF in figure 6.2, respectively).

Conversely, the right plot depicts the total number of parkings recorded in each zone. It clearly shows that areas where the majority of rentals/parkings occur correspond to zones downtown where people frequently drive, drop the car, and then someone else re-rent the same car.

In a nutshell, in busy areas, the average parking time is short, and the number of rentals and parkings recorded is high. This reflects the specific usage of FFCS according to which cars move to downtown areas in the morning, then are rented to move within central areas, and finally are driven back to the periphery at the end of the day. Similar results apply to all cities – see Fig. 6.3b which details Vancouver statistics.

In the following, I leverage this knowledge obtained by actual data to compare the design of and optimize different charging station placement policies.

Takeaway: Periphery zones are characterized by a long parking time, while central areas are characterized by many parkings which last a short time.

6.4 Impact of charging station placement

6.4.1 Simulation setup

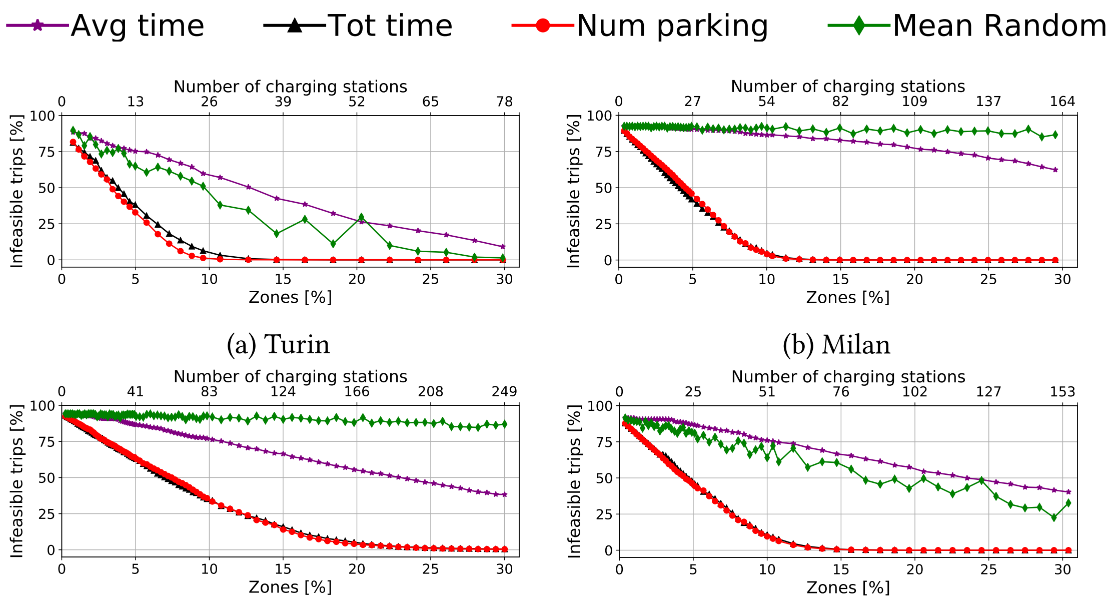
In this section, I run several accurate simulations through the simulator described in chapter 5. More in detail, for each city, I built the trace replicating the start and end for each rental, with a fleet size reported in table 6.1. Those particular simulations deeply studied the pro and cons of different provider settings like charging station placement and car return policy.

In particular, the main provider side performances is the percentage of *infeasible trips* (trips where the battery completely run out before to reach the destination). From a user point of view, I measure the *discomfort metrics*, the metrics that may be seen as extra work from the user side. They namely are the *charge events*, *percentage of reroutings*, *average plug time* and the *average walking distance.*

The best configuration is the one that makes the *infeasible trips* close to 0% and minimizes all the other metrics.

6.4.2 Results

I consider; initially, the Free Floating return policy (described in section5.4), and I study the impact of different charging station placement policies. The aim is to check what



(a) Turin (b) Milan

(c) Berlin (d) Vancouver

Figure 6.4: Percentage of unfeasible trips as function of charging station number, for different placement algorithms and city. Placing charging stations where cars are frequently parked is much better than where cars stay parked for long time.

would be the minimum number of charging stations to install to sustain an FFCS system based on electric vehicles that are equivalent to the one currently in use.

Figure 6.4 shows the performance of the different placement algorithms in terms of percentage of infeasible trips with respect to the number of charging stations N for each city. Each charging station has $k = 4$ poles. The bottom x-axis reports the percentage of equipped zones with respect to the total, while the top x-axis reports the actual number, different for each city.

It is possible to observe notably different performance for different placement algorithms. First, the average parking time placement policy (*Avg time* - purple line) has very poor performance in all the cities. Even a simple random choice sometimes performs better (*Mean rnd* - green line, obtained as the average of 10 random instances). However, in Milan – figure. 6.4b – and Berlin – figure 6.4c, the random placement results the worst. This is due to the larger number of zones, which makes the space of available solutions much larger.

Second, the total parking time (*Tot time* - black line) and the total number of parkings (*Num parking* - red line) perform similarly and consistently better than other policies. A 10% coverage in Turin, 11% in Milan, 23% in Berlin%, and 13% in Vancouver lead to about 2% of infeasible trips. In all the cities but Berlin, it is possible to reach a negligible percentage of infeasible trips with just 15-18% of charging zones. Instead, in Berlin, it is possible to still have some infeasible trips with 30% of charging zones. Recalling that with the current car model it is possible to travel 135 km 5.5, the presence of infeasible trip is explained by looking at the rental distance presented in 6.2. Trips in Berlin can be as long as 39 km. Therefore, with only 4 long-trips that do not end in a charging station area, the battery could run out the energy.

The overall trends confirm the intuition of why the recharging stations placement algorithm is of primary importance. *Avg time* placement favours peripheral zones where few trips end, and where cars stay parked for long time, sometime longer than the time required for a complete charge, (see left heat maps figure 5.1c and figure 5.1b). On the contrary, *Num parking* and *Tot time* favour city center areas, where cars are frequently parked for short time (see right heat maps in figure 5.2c and figure 5.2b).

Figure 6.5 confirms this intuition. Indeed the *Avg time* placement generates much longer plugged times, often much longer than the time needed for a full charge. Therefore, many cars occupy the charging poles when they are already charged, preventing other cars from using those poles and increasing the number of infeasible trips.

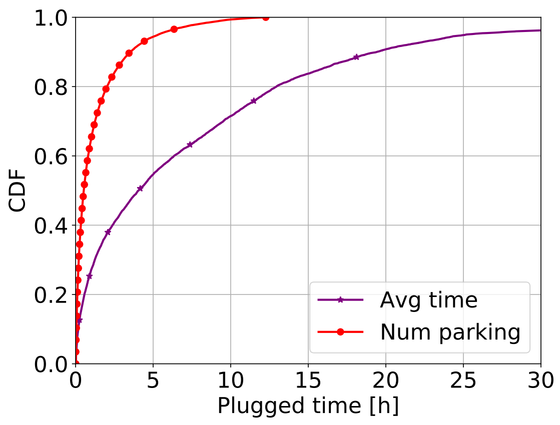


Figure 6.5: CDF of the time spent by a car at a charging station (Z=40), for *Num parking* and *Avg time* placement algorithms in Turin.

Takeaway: Placing charging stations in areas where cars stay parked for long time is not convenient. Placing charging stations in areas which allow many cars to recover the (little) energy consumed in the (short) trips results in a much better policy.

Given this, the total number of parking is the placement algorithm, and it will be used for the rest of the chapter.

6.5 Impact of return policy

I now investigate the impact of the different return policies described in section 5.4. In particular, I quantify the implications of asking customers to return the car to a different zone than the desired one when the battery is below a critical level.

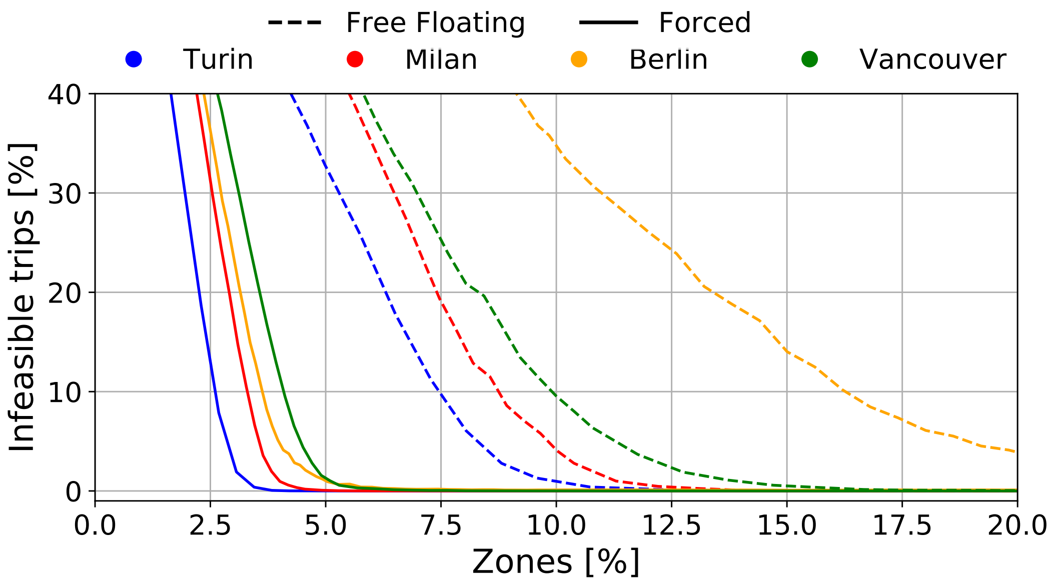
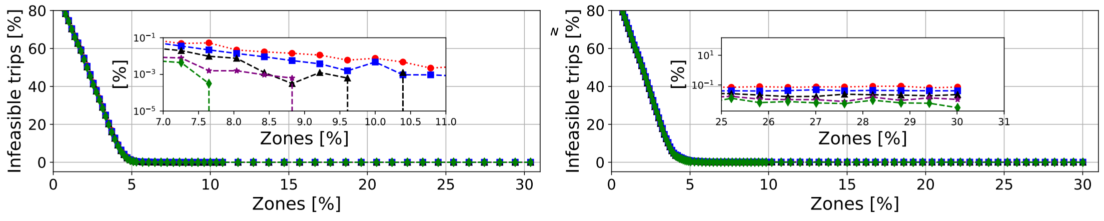


Figure 6.6: Infeasible trips when comparing the Free Floating vs Forced return policies. Forcing customer to charge when $c < \pi$ drastically improves performance.



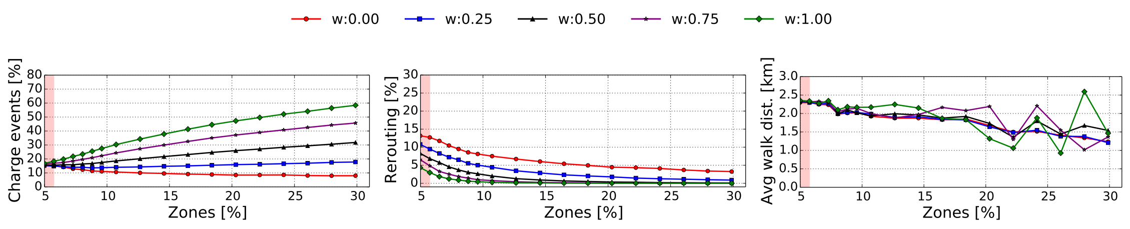
(a) Vancouver

(b) Berlin

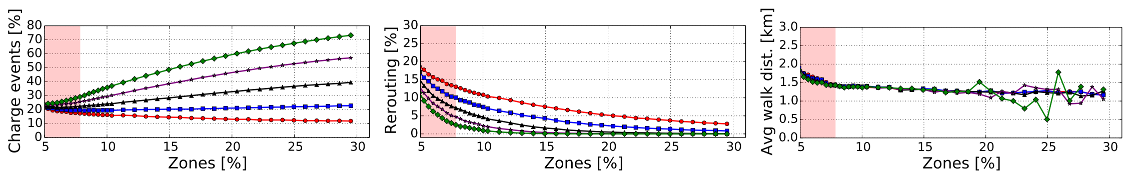
Figure 6.7: Impact of customer willingness to cooperate w. Albeit the small impact, the higher w is, the better it is.

6.5.1 Impact on infeasible trips

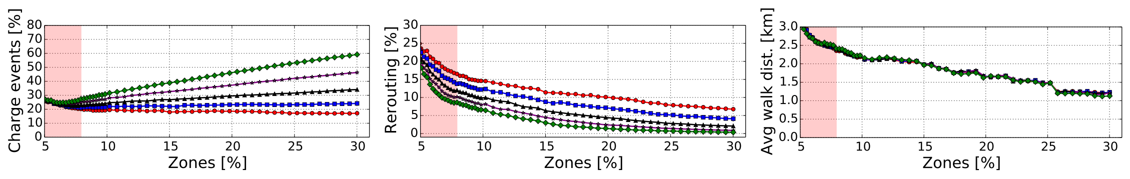
Figure 6.2 has already shown that trips are typically limited. This is instrumental in choosing a proper minimum charging threshold of π. In particular, π must guarantee to cover the maximum trip distance and the corresponding energy being consumed. For instance, a maximum distance of 20 km corresponds to about 15% of battery capacity for the considered car model. In the following, I take a conservative approach and set the minimum battery charge threshold $\pi = 25\%$. To make results comparable, I keep the same threshold also for Berlin, where the maximum traveled distance grows to 39 km, i.e., suggesting $\pi \geq 30\%$. Here the choice is not conservative.



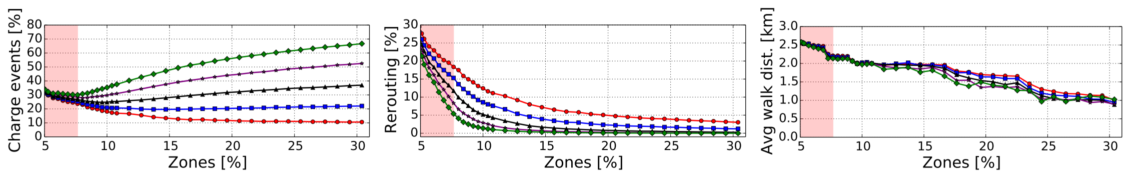
(a) Turin



(b) Milan



(c) Berlin



(d) Vancouver

Figure 6.8: Charge events percentage (left), re-route events percentage (middle) and walk distance averaged over re-routed trips (right), per city. Increasing w benefits the customers' experience by reducing the re-routing events significantly, but increasing the charging events.

As before, I focus on the percentage of the infeasible trip with respect to N. I compare the results for the Free Floating and the Forced policies. Figure 6.6 shows the results. The Forced policy (solid lines) performs much better with respect to the original Free Floating policy (dashed lines). In a nutshell, adopting a policy that mandates customers to charge the car when the battery level gets below a threshold drastically reduces the number of infeasible trips, even with a handful of charging stations. Indeed, in all cities is present a negligible percentage of infeasible trips ($< 0.1\%$) with more than 8% of zones equipped with charging stations.

I now focus on the impact of the willingness w in the Hybrid policy. I want to understand if the more altruistic the customers are, the higher are the benefits for the system.

Figure. 6.7 details the percentage of infeasible trips with a different willingness probability for Vancouver and Berlin. Turin and Milan are similar to Vancouver and not reported for the sake of brevity. The figures show that increasing w has very little impact on the percentage of infeasible trips. Only by looking at the insets that offer a zoom in log-scale, it is possible to observe that an increasing willingness reduces infeasible trips, which are however, already a marginal percentage of trips. This is due to the higher average battery level, obtained by supposing the Free Floating policy on the not of the Forced one. In Berlin, some infeasible trips are still present even when 30% of zones are equipped with charging stations. This is due to the maximum length of the trips, confirming the need to increase the threshold π.

Takeaway: Asking customers to return the car to a charging station when the battery level goes below a minimum level drastically improves system efficiency. With just 8% of the zones covered by charging stations, the systems become in all the cities almost self-sustained.

6.5.2 Impact on customer experience

As there is no strong evidence that the overall system would perform better with customers' altruistic behavior, here I check benefits on the customer experience. Forcing a customer to park in a charging station can be annoying because the customer has to reach the charging station and lose time to plug and unplug the car to and from the pole. Even worse, rerouting customers to other zones for charging increases the distances they have to walk to reach their desired destination. In the following, I measure the customer's experience through the KPIs described in 5.5.

Figure. 6.8 reports, for each city, and for different willingness the percentage of charge events (left plots), the percentage of rerouting events (middle plots), and the average walk distance when rerouted (right plots). In all graphs, the shaded area highlights the infeasible region, i.e., when infeasible trips are higher than 0.1% in at least a case. The lack of charging zones creates artifacts here.

Focus first on the charge percentage - leftmost plots. When the number of charging stations is close to the minimum, most poles are occupied by cars that have been forced to charge. This leaves little room for opportunistic charges, and there is little impact of w. For the increasing number of charging stations, the opportunity to find a free pole in $o(i)$ increases. Thus, the higher is w, the higher are the recharging events. Correspondingly, the average battery charge increases – see Fig. 6.9a.

Interestingly, the charge percentage decreases for a selfish customer ($w = 0$), reaching about 5-8% for the sufficiently large number of charging stations. This corresponds to the average number of charges per car that guarantees a minimum battery charge of 25%. Indeed, given the average rental distance of 5 km (middle ECDF in figure 6.2) and a battery range of 100.5 km (at the net of the safety threshold π), a car could sustain on average 20 normal trips before needing to charge.

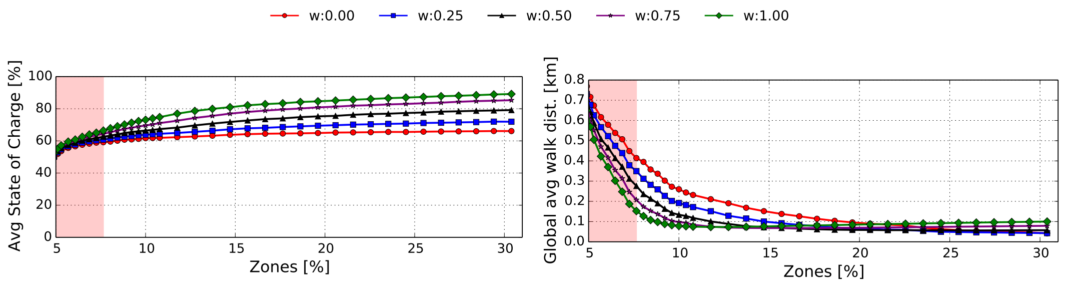
Move now to the percentage of re-routing events - middle plots in figure. 6.8. Two important effects are visible. First, rerouting probability decreases as expected: the more the stations are, the more likely customers find a charging station at their desired final zone. With selfish behavior ($w = 0$), the fraction of rerouting events remains large even for large N. Second, the more collaborative customers are, the better it is for the entire system. With half of the customers that return the car voluntarily to charging station if present in their final destination, for Milan, Turin, and Vancouver, the re-routings are less than 1% with 18% of charging zones. These can be handled by a relocation policy, i.e., the system could take care to charge those less than 1% of cars whose battery level is close to π. Note that this corresponds to a maximum of 53 relocation events per day. Instead, for Berlin, the number of rerouting remains larger than 3%, mainly due to its larger size.

At last, focus on the average walk distance when the car is rerouted – rightmost plots in figure. 6.8. When forced to charge to a different zone than the desired one, customers would be asked to drive far, a likely unacceptable penalty unless mitigated by offering incentives to customers, e.g., offering a free rental when re-routed.[3] Notice that by increasing the number of charging stations, the walked distance is reduced, but not linearly. This is also due to the fact that charging stations are not placed uniformly in space, following the number of parking heuristic.

Therefore, I would like to have $w = 1$ to reduce (far) re-routings and infeasible trips, and $w = 0$ to reduce charging events. In an attempt to take into consideration both aspects, I compute the (global) walk distance averaged overall trips. This considers the penalty due to (i) the rerouting events and (ii) the walk distance when charging in a pole of the desired final destination (pole distance would be 150 m, on average). Figure. 6.9b reports this general average walk distance for Vancouver. For more than 10% of zones with charging stations, with all the policies, customers have to walk on average less than 400 m. When the number of charging stations is low, increasing w reduces the average walking distance since opportunistic charges reduce rerouting events. However, increasing the number of charging zones increases the cost of always driving to a pole in the same zone. In the case of Vancouver, after 23% of zones, the best policy switches from $w = 1$ to $w = 0$. Therefore, the policy to use may be different according to the number of charging stations. Overall, for all the four cities and 10% of zones, and choose the policy with $w = 1$, customers on average walk less than 200 m to reach their desired destination.

Takeaway: Hybrid policy significantly reduces the number of times the customer has to drive to a charging station in a different zone than the desired one. However, it increases the number of times the customer parks at a charging station and has to plug the car into the pole. Therefore, one must be cautious when weighing these results and designing the return policies which impact the customers.

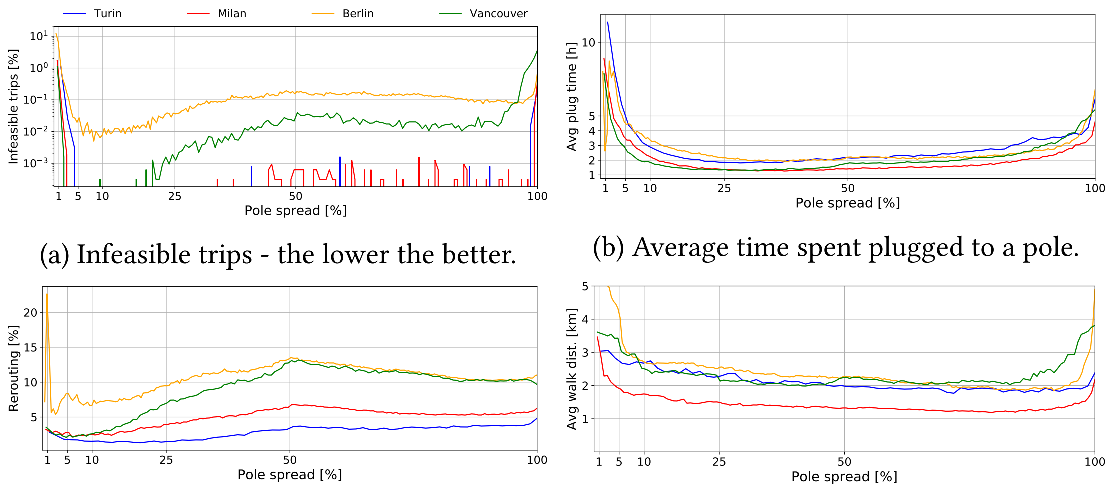
[3]The noise for large N is due to the very small number of rerouting events.



(a) Average State of Charge - the higher the better.

(b) Global average walk distance - the lower the better.

Figure 6.9: Details of the average battery charge status and global average walk distance for Vancouver. Other cities have similar results.



(a) Infeasible trips - the lower the better.

(b) Average time spent plugged to a pole.

(c) Re-route events percentage - the lower the better.

(d) Average walk distance for re-routed trips - the lower the better.

Figure 6.10: Impact of pole distribution among zones. Concentrating all poles in very few hubs performs poorly, as well as placing single pole charging stations.

6.6 How to distribute poles in stations

In the previous sections, I assumed charging stations with $k = 4$ poles each. Here, I study the impact of installing the charging station with a larger or smaller number of poles each. I keep the total number of charging poles $K = kN$ constant and equally distribute them in a varying number of charging stations N. In other words, I check if it is better to have (i) big charging hubs with many poles (one single hub corresponds to $N = 1$, with K poles) or (ii) a very large number of charging stations, each with one pole ($N = K, k = 1$). I call *pole spread percentage* the percentage of zones in which poles are distributed among, i.e., $100N/K = 100/k$. For example, pole spread percentage 5 corresponds to 20 poles per zone ($k = 20$), spread percentage 10 corresponds to 10 poles per zones, etc., up to spread 100 that corresponds to a single-pole per each charging zone ($N = K$ and $k = 1$).

For each of the four cities, I pick a constant number of charging poles K, corresponding to a percentage of 7% of charging zones when $k = 4$.[4] Then, I distribute the poles evenly among different numbers of N zones, chosen according to the highest number of parkings (as in the previous Sections). I consider the Hybrid policy with $w = 0.5$ and simulate the resulting system.

From top to bottom, figure 6.10 reports the percentage of infeasible trips; the average time cars are plugged into a charging pole (even if they are completely charged); the percentage of re-routed trips, and the average walk distance for re-routed trips. Colors refer to different cities and the x-axis reports the spread percentage. Note that with $k = 4$, as in the previous experiments, we have a spread percentage of 25% for all the cities.

Focus first on 6.10a. With spread percentage going below 5%, hence concentrating poles in very few hubs, the number of infeasible trips quickly grows to non-negligible values. Even Turing and Milan suddenly suffer of a sizable percentage of infeasible trips. The lowest values are obtained with a spread between 5 and 20, meaning that increasing the number of poles per charging station in the $[5 - 20]$ range helps the system to sustain. For instance, placing stations with $k = 6$ poles lets Vancouver sustain all trips. Conversely, spreading a lot of charging stations, each equipped with one pole, is also not optimal. Poles are occupied for a long time by cars that customers tend to not rent since located in non-popular areas, as can been seen by the (too) high plugged time in figure 6.10b.

figure 6.10c shows the percentage of re-routings. Here I also observe that it is better to have a low spread percentage. For the region where there are negligible infeasible trips (spread between 5 and 20), the percentage of re-routings increase both because poles start to be located in a less popular destination and because cars are charged for less time (see figure 6.10b).

Lastly, I show walk distance for re-routed trips in figure 6.10d. As expected, the walk distance slowly decreases with the spread percentage. Indeed, when customers are re-routed, they can return the cars in more areas, likely closer to their desired destination. Only when single poles are used, the walk distance increases again. This is justified by cars that stay attached to poles for (too) long time, reducing the availability of free poles, thus forcing customers to drive further away.

Therefore, the best trade-off is hard to detect and depends from city to city. In general, it looks better to concentrate poles only in those zones where cars are frequently

[4]Each city has a different K, producing a different number of infeasible trips.

rented and returned so as to increase the chance to find a free pole and let the battery quickly charge before the next rental makes the pole free again.

Note that both extreme solutions of pole spread percentage would also cause the highest installation and operating expenditures. The single hub solution would require to have a huge amount of power at disposal in a single location. In comparison, the single-pole solutions would largely increase the installation cost and the occupied road section.

Takeaway: Choosing the number of poles per zone must consider different factors. Concentrating all charging poles in very few hubs or spreading them among all city areas performs badly. The intermediate solutions look beneficial and must be carefully weighted, also considering the cost of installing charging stations.

6.7 Conclusions

Designing an electric vehicle free-floating car-sharing system leads to many interesting problems and trade-offs. In this chapter, I built on actual rental traces to study via accurate simulations the impact of i) the charging station placement and ii) return policies. Considering charging station placement, I demonstrated that it is better to place charging stations within popular parking areas (e.g., downtown), even if the parking duration is short.

The analyses have shown that an FFCS solution with electric vehicles can almost sustain itself, even with very few charging stations (8-10% of city zones). These results are also obtained thanks to customers' collaboration by returning the car to a nearby charging station or whenever the battery level drops below a target threshold.

Car sharing providers shall consider the trade-off between usability, costs, and benefits for the customers. The results hint at possible alternative design solutions, i.e., adopting simple relocation policies that would move cars that need a charge only, a promising solution to limit discomfort for customers due to re-routing enforcement. This could be achieved by considering giving incentives to customers.

Chapter 7

Data-Driven Optimization of Charging Station Placement in Electric FFCS

This chapter is mostly taken from *"Data driven optimization of charging station placement for EV free floating car sharing"* published in the 21st International Conference on Intelligent Transportation Systems in November 2018 ([58]) and *"Free floating electric car sharing design: Data driven optimisation"* published on Pervasive and Mobile Computing journal in April 2019 ([46]). I am the main contributor of this paper while other authors supervised the entire publishing process.

7.1 Introduction

Mobility and pollution are challenging problems in our cities. Private vehicles, still important urban transportation means, are among the major contributors to both congestion and air pollution. Because of this, smart and shared mobility are seen as a key component to reduce emissions and traffic [27]. Given a fleet of cars, Free Floating Car Sharing (FFCS) systems allow customers to pick and drop a shared car everywhere inside an operative area, thus reducing the number of private cars and increasing the number of available parking spots. The conversion from internal combustion cars into Electric Vehicles (EVs) is seen as the next big opportunity to drastically reduce pollution inside urban areas [52]. However, the design of a system based on electric cars entails the deployment of a charging station network [59].

In this work, I tackle the design of an electric FFCS system. This is a challenging problem, given charging constraints which impact car availability, and the cost of the infrastructure setup and maintenance. The design of the charging station infrastructure requires thus ingenuity to maximise customers' comfort, and minimise cost for the operator [49, 50, 51].

Two are the main problems that need to be faced: i) the charging station placement problem, i.e., how many and where to install charging stations; and ii) the return policy customers have to follow at the end of the rental, i.e., in which cases to ask the customer to return the car to a charging station. In this chapter I face both the above problems. Notice that the number of charging stations is directly related to system installation costs.

Data is fundamental to answer these questions. While in the past some works have proposed solutions for the design of electric FFCS [52, 53], this work is among the first to take a complete data-driven approach [49, 50, 51, 54, 60, 61] in an electric FFCS.

I start by leverage the data collected from more than 2 months of rentals in 2017 with the software described in chapter 2. I remind to the characterization done in sections 3.3 on Turin and 5.3 studying in detail Turin, Vancouver and Berlin.

Then, relying on the simulator described in chapter 5, I replicate the exact same events recorded in the traces to accurately mimic actual customers' habits. It simulates the usage of each EV, its battery consumption and charging, while considering different design parameters. With this, I run thorough simulations to understand the implication of the design choices, such as charging station placement algorithms, and car return policies.

Reminding that section 6.4 shows that placing the charging stations in those areas where cars stay parked for long periods performs worse than a totally agnostic random placement. Instead, placing charging station in those areas where cars are frequently parked even for short periods guarantees better performance.

Starting from that I gauge the benefits of considering collaborative car return policies, where customers voluntarily or forcibly return the car to a charging station in case the battery level decreases below a threshold like the authors of [62] proposed. This kind of return policies are inspired by the user-relocation model presented in [63, 64, 65, 66], where the relocation is driven by the presence of parking and charging stations and not by the demand areas. It is possible to observe that this halves the number of charging stations required to sustain the system. However, this increases customer's discomfort, in terms of number of times customers have to return the car to a charging station, at the cost of additional distance from their desired final destination.

To solve this tension, I further optimise the placement of the charging station by means of global optimisation algorithms. I, implement and validate two algorithms: a hill-climb local search and a genetic algorithm, both tuned to minimise system cost and customer's discomfort. Results are surprising: just equipping 5% of the city area with charging stations guarantees the system to self sustain, with no cars ever running out of battery in two months of trips. This corresponds to install only 13 charging stations in the whole city of Turin, which has 1 million inhabitants. Furthermore, the placement found by the genetic algorithm guarantees only 4% of re-routing events, with the customers parking the car, on average, within 90 m from their desired final destination.

I believe results presented in this paper, guided by actual usage pattern for FFCS customers, are very important for regulators, policy makers, car sharing providers, as well as for researchers working in this area. This data driven approach provides novel opportunities to guide the design of electric car sharing system, where the realistic figures provided by data allow investigating solutions that meet both customer requirements and limit system costs.

After discussing related work in Section 7.2, I present the methodology I adopted to collect data and characterise the data-set in Section 7.3. In Section 7.4 I describe the simulation model, its parameters and metrics of interest. Section 7.5 presents the different placement heuristics and the algorithms I design to optimise the placement. Section 7.6 discusses the impact of simple charging stations placement policies and return policies, while Section 7.7 reports the results of the optimisation and their validation. Section 7.8 discusses limitations and future work, before drawing conclusions in Section 7.9.

7.2 Related works

The diffusion of the free-floating approach to car sharing led to increasing attention by many researchers, with many analyses of these systems and their extension to electric vehicles. The studies performed in 2011 by Finkorn and Müller [27, 55] are the first attempts to analyze the benefits of FFCS for the population. Their results on customers' characterization, like traveled distances and rental duration, are similar to what was pointed out in previous chapters.

Later works [56, 29, 22] also collected data and analyzed the mobility pattern of customers and differences among cities. While providing insights on usage patterns, these works do not discuss the implications on Electric Vehicles based FFCSs. I also introduced UMAP, described in chapter 2, a system to harvest data by crawling FFCS websites. Here I use, again, the traces collected with UMAP to drive the system design.

The introduction of EVs for private and public transportation brought the problem of the design of the electric charging station infrastructure. After a survey among FFCS customers in Ulm (Germany), authors of [52] investigated the positive influence and feasibility of an electric FFCS system. Authors in [54] show the benefits of placing charging stations with different capacity according to the car parking duration. Authors of [67] presents a simulation study similar to ours but using random models to generate random trips rather than actual traces. Their algorithms tend to place charging stations along frequently used streets, so to let drivers top up the battery in 10 minutes.

Few data-driven studies address the charging station placement, by respectively minimizing the cost of installation, power loss and maintenance [48, 49, 51], or by minimizing the customers' walked distances necessary to reach a charging pole [50]. In [49], authors study the impact on the power distribution grid, with limited focus on FFCS performance. Authors of [51] instead of focus on charging station design to minimize customers' anxiety. With compared to chapter 6, where I presented a study of charging station placement based on actual data, while here I build upon this work, a more in-depth study that includes global optimization algorithms, never considered before.

Other works focus on station-based car sharing systems. Authors of [60] present algorithms to place the parking stations in a two-way scenario. They consider a combustion engine fleet and solve the problem by considering real data from operative car sharing systems. The same authors propose a similar methodology considering a one-way scenario and electric vehicles [61]. Here they use synthetic data and other socio-economic information to estimate the demand. Both works have similar goals but are limited to station-based car sharing.

Considering return policies, interesting data-driven research is presented in [62]. The authors focus on the station-based car-sharing system with electric vehicles, finding that the best policy is to charge a car only when its state of charge goes below t.a minimum threshold. This is similar to the return policies I consider in this paper.

Other works focus on FFCS with EVs to study the revenue considering a demand-supply scenario for energy [68], introducing policies to free charging stations when occupied by fully charged cars [53], maximizing revenue by moving cars in areas of high-demand [66], or providing incentives to customers to balance fleet [63, 64, 65]. These works are orthogonal to the one presented in this chapter.

To the best of my knowledge, this work is among the first to take a completely data-driven approach for designing an electric FFCS system by optimizing different metrics impacting the customer experience.

7.3 Methodology

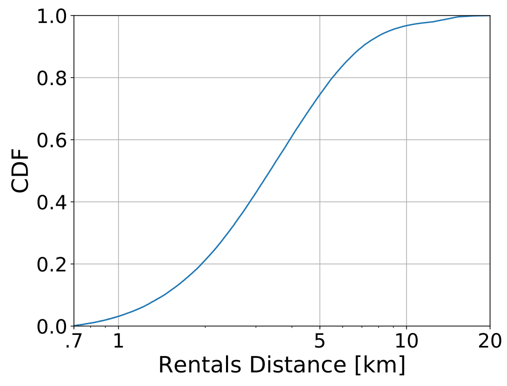
In this section, I briefly recall the data collection pipeline and the simulation scenario adopted in this chapter, which basically replicates what I described in chapter 6.

7.3.1 The dataset

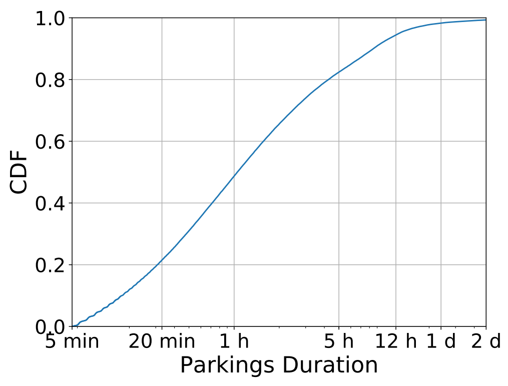
The dataset I use for the following analyses is collected through the software described in chapter 2. The subset sampling is the same performed in section 6.3. Figure 7.1 recalls the main features of collected trips. In particular, in Turin, the 95% of rentals covers less than 10 km, and the 95% of parking lasts less than 12 hours.

Now, I recall the spatial users' behaviour, widely described in chapters 3, 4 and 5. Figure 7.2a shows the heatmap of the total number of parkings in each city zone. The warmer the color is, the more frequently cars are parked here. The hot areas correspond to the city center, which exhibits the highest number of parkings. Customers rely on car sharing for traveling and moving downtown, a working area full of shops and restaurants. The zones with more parkings are close to the train stations, where 47 parking events per day are observed on average.

Figure 7.2b shows the heatmap of the average parking time for each zone. Peaks are on the borders of the operative area, where parking events last more than 24 hours.

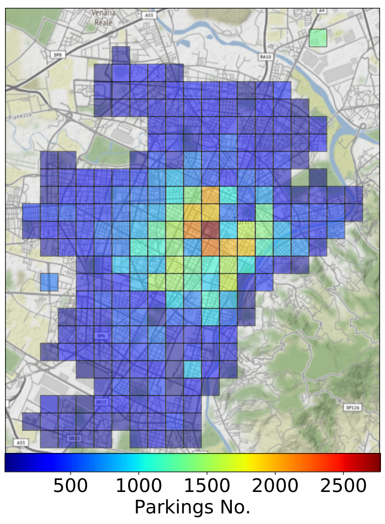


(a) CDF of travelled distances. X-axis is logarithmic.

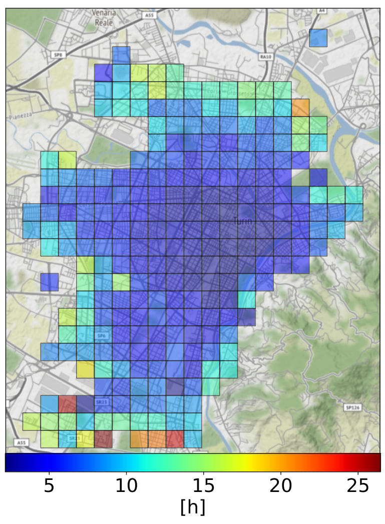


(b) CDF of parking durations. X-axis is logarithmic, and limited to 2 days.

Figure 7.1: Characteristics of the trips in our data-set.



(a) Number of Parkings.



(b) Average Parking Time.

Figure 7.2: Heatmaps showing (a) number of parkings per zone and (b) average parking time per zone. Warmer areas have larger values.

Few cars reach these peripheries and rest unused for a long time (see also rightmost part of Fig. 7.1b). The lower values are registered in the downtown, where cars stay parked only for 85 minutes on average.

7.4 Simulation scenarios

The goal is to study different design choices for electric car-sharing systems based on collected data. For this, we developed a flexible event-based simulator that allows us to compare different algorithms and tune parameters while collecting metrics of interests.

7.4.1 Simulation parameters

Through the simulator widely described in chapter 5, I simulate a fixed fleet of electric cars. Each car is characterized by its parking location and the current status of the battery charge. The simulator takes as input the pre-recorded data-set of rentals, i.e., the trace, characterized by the start and end time, and initial and final geographic coordinates.

Depending on the return policy, the customer may connect the car to a charging pole. , namely *Free Floating*, *Needed* and *Hybrid*. The *Free Floating* policy never obliges the customer to bring the car far from the desired ending location, even in case the battery charge is close to exhaustion. *Needed* mandates to connect cars to a charge station only if the battery runs low, thus trying to protect from battery exhaustion. *Hybrid* mixes the two policies letting customers opportunistically recharge the battery whenever they park close to a charging station. More information in section 5.4.

Notice that policies similar to *Needed* have been introduced in [62], where the system make the users charge the car considering the battery state of charge, the instantaneous electricity cost, and the user's range anxiety.

7.4.2 Key Performances Indicators

I measure the following metrics, defined in section 5.5 that I identify as having an influence on the customers' quality of experience:

- *InfeasibleTrips%*: percentage of infeasible trips due to completely discharged battery observed during the whole simulation;
- *Charges%*: percentage of trips where the customer connects the car to a charging pole, implying the burden to plug the car;
- *Reroutings%*: percentage of trips where the customers are rerouted to a zone different from their original destination because they are forced to charge the car;
- *WalkedDistance*: the walked distance from the desired destination.

Infeasible trips are critical, and the system shall be engineered so that they never happen. Other performance metrics shall be minimized. In addition to the above metrics, the simulator collects statistics about car battery charge level and the fraction of time a battery stays undercharge.

The key design parameters that I focus on are (i) the number of zones Z which are equipped with a charging station; (ii) the locations of charging stations within the city; (iii) adopted return policies.

I consider the following scenario: the fleet has a constant number of cars equal to 377 (the same as observed in the trace). Electric cars have the same nominal characteristics as the Smart ForTwo Electric Drive, i.e., 17.6 kWh battery, for 135 km of range, with a discharge curve that is proportional to the traveled distance (12.9 $kWh/100\ km$).[1] Charging stations have 4 low power (2 kW) poles each. These are cheap to install and a good compromise between costs, power requested, and occupied road section. I model a simple linear charge profile (complete charge in 8 hours and 50 minutes in the case of study).

7.5 Strategies for charging station placement

The main objective of this work is to assess what is the best charging station placement. Assuming a total of Z zones and N charging stations, there are $\binom{N}{Z}$ possible placement solutions, which makes it prohibitive to find the optimal solution exhaustively. For this reason, I evaluate different approaches. The first one uses domain knowledge acquired by characterizing the data about current usage to propose heuristics. The second, instead, uses two different data-driven simulation-based optimizers.

7.5.1 Heuristic placements

Recalling the tiling procedure described in chapter 5, I divide Turin into cells, computing in each cell the likelihood l_z that measures spatial users' pattern usage in each zone. This work takes in consideration the *Total number of parking*, *Average parking time* and *Random placement*. Definitions and more details in are available in section 5.3. Figure 7.2 quantitative depicts the distribution of those likelihoods.

The first two heuristics are driven by the intuition to place charging stations in those zones where cars are likely to be parked for a long time or frequently. The latter is presented as a baseline for comparison. Referring again to Figures 7.2a and 7.2b, the charging stations will be located in the zones with warmer colors, respectively for the total number of parkings and average parking time policy.

7.5.2 Simulation based advanced optimisation strategies

Given the complexity of the optimization problem and the humongous space of possible solutions, I investigate the adoption of meta-heuristics, a class of global optimization

[1] https://www.smart.com/uk/en/index/smart-electric-drive.html

algorithms [69]. These algorithms explore the space of possible solutions in smart ways, looking for better solutions while avoiding getting trapped into local minima.

In this case, the evaluation of a solution requires the simulation of two months of rentals, which is performed in approximately 5 seconds on a high-end machine. It is then important to consider that I have limited resources in the choice of meta-heuristics to consider. The class of optimization problems where the number of solutions (i.e., fitness evaluations) have to be limited as much as possible lies in the so-called expensive-optimisation [69, 70].

In the literature, there are several architectures and algorithms suitable for global optimization in tough numerical problems [70], with direct-search class algorithms that explicitly target expensive-optimisation problems.

In this work, I consider a simple local-search algorithm based on a hill-climb method and a more complex and powerful genetic algorithm. I explicitly design both algorithms with the perspective of reducing the number of simulations.

I consider the single-objective case, i.e., algorithms have to minimise a single fitness function f defined as follow:

$$f = M \cdot InfeasibileTrips\% + WalkedDistance$$

As commonly done in linear programming, M is a number big enough to make the first addend always larger than the second. In this way, *InfeasibleTrips%* has to be minimized first. Secondly, the algorithms start minimizing the *WalkedDistance%*. In a nutshell, the algorithm looks for solutions that make all trips feasible, and only then it targets the customers' discomfort, i.e., reducing the walked distance. As he next section will show, reducing *WalkedDistance* will naturally help in reducing also *Charges%* and *Reroutings%*. Indeed, in its definition, the walked distance weights both these metrics.

Hill-climbing local search

Hill-climbing methods belong to the family of local search algorithms and are a popular choice because they are fast, simple to implement, and requires limited computational resources. They are iterative algorithms that start with an arbitrary solution, then attempt to find a better solution by making incremental changes to the solution. If the change produces a better solution, it is selected as the current solution. Incremental changes are then made to the latter until no further improvements can be found. For non-convex problems, like the one here faced, these methods will find only local optima, from which it would be impossible to escape. Local optima are not necessarily the globally best possible solution.

This implementation of a hill-climbing algorithm is very similar to the coordinate descent version [71]. The algorithm starts from the best configuration found among the three heuristics. At each iteration, the algorithm randomly picks a charging station and moves it in an empty neighboring zone, i.e., north, south, east, and west adjacent zones. All other charging stations are left untouched. If there is a direction of improvement, it performs a line search along the best direction, i.e., it keeps moving the same station in the same direction. When no improvement is possible, the algorithm takes another charging station at random and try to move it as before. When no improvement is found after a complete cycle of all charging stations, a local minimum is reached, and the algorithm exits. A Maximum number of visited solutions stop condition is present too.

In this implementation, I check multiple neighbors in parallel. Moreover, the algorithm keeps the memory of all tested configurations, hence avoiding useless and expensive simulations. In the experiments, on average, the optimization reaches convergence within 1 500 maximum tested solutions.

Genetic optimiser

Genetic algorithms are a particular class of evolutionary algorithms inspired by the natural evolution [72]. They are known to work well when dealing with discrete variable functions, as in this case.

Genetic algorithms start creating a random population of a given number of individuals. In this case, each individual corresponds to the random placement of charging stations. A mating pool is created from the initial population, then the offspring is generated by crossover and mutation operations. Crossover mixes the genes from different individuals picked at random, i.e., a child is created from the union of the genes of both parents. To keep the number of genes (i.e., charging stations) constant, random genes are removed so that at the end, Z are left. Mutation instead moves a single charging station in a random empty zone. During crossover operation, some genes may mutate with low probability ($P\{mutation\} = 0.02$ in this case).

The presence of clones is avoided by the algorithm, which discards the copies, thus encouraging the exploration of the search space and saving precious resources. The algorithm estimates the quality of each new individual computing the fitness function f, i.e., by running an entire simulation.

As by natural selection, the best individuals survive to the next generation, while the worst individuals are suppressed. The optimization loop continues until the maximum number of generations is reached (200 in this case).

If the diversity of available genes, i.e., the total number of distinct charging stations in the whole population, decreases too much without improvements, it is triggered an increase of the population genetic diversity by increasing the mutation probability ($P\{mutation\} = 0.2$ in this case). This randomizes the evolution.

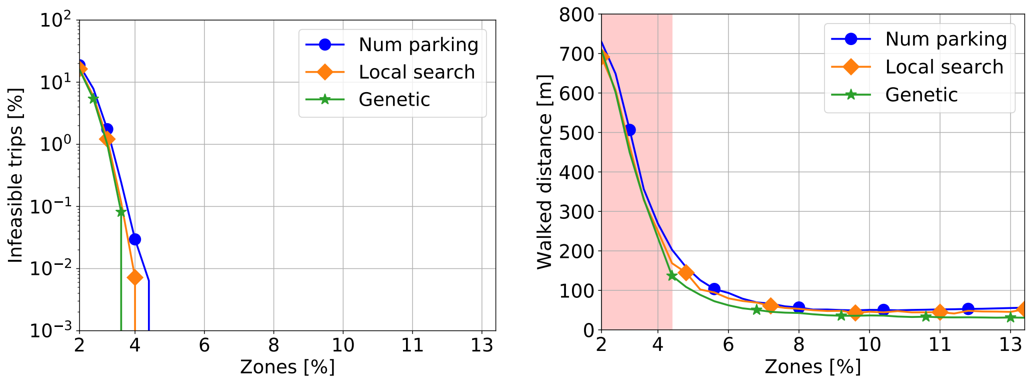
The algorithm is amenably suitable for parallel implementation since the fitness of each individual belonging to a specific generation can be analyzed separately from the others. Another advantage of genetic algorithms is the widespread exploration of the solution space, while local search algorithms tend to explore only a limited portion of the space.

In the experiments, I set the initial population to 100, with 50 new individuals created at each generation by crossover. Most of the optimizations ended within 100 generations, i.e., 5 000 overall solutions are evaluated through simulation.

7.6 Meta-heuristic optimisation of the charging station placement

In section 6.4, I pointed out how *Num Parking* placement heuristic works better than the other two. Weighting also charges and rerouting, the *Hybrid* policy shows better performances than *Needed.* For this reason, in this section, I focus on the *Hybrid* return policy with charging stations covering less than 15% of the zones. I further optimize this scenario by running the meta-heuristic placement algorithms and comparing the results with the *Num Parking* placement. Optimisation with *Needed* return policies are briefly discussed in 7.7, where very similar results are obtained.

The hill-climbing local search, here abbreviated in *Local Search*, uses *Num Parking* placement as initial solution. The *Genetic* algorithm creates a totally new solution without exploiting any previous knowledge. Recall that both algorithms are designed to find the best charging stations placement that guarantees 0 infeasible trips and to minimize the overall distance the customer has to walk to reach the final destination.

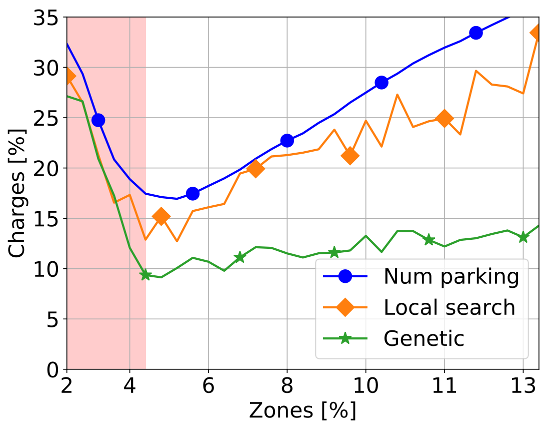


(a) Percentage of infeasible trips. Y-Axis is logarithmic.

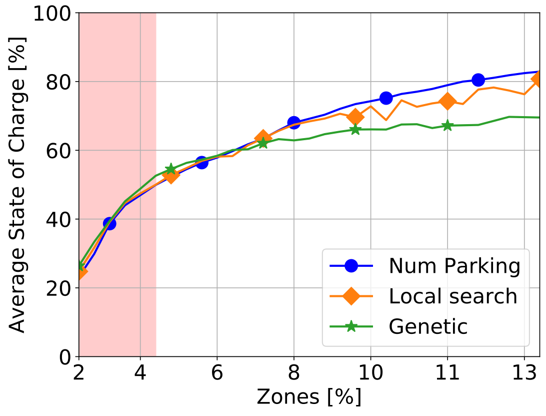
(b) Average walked distance.

Figure 7.3: Objective metrics to minimise - with *Hybrid* return policy.

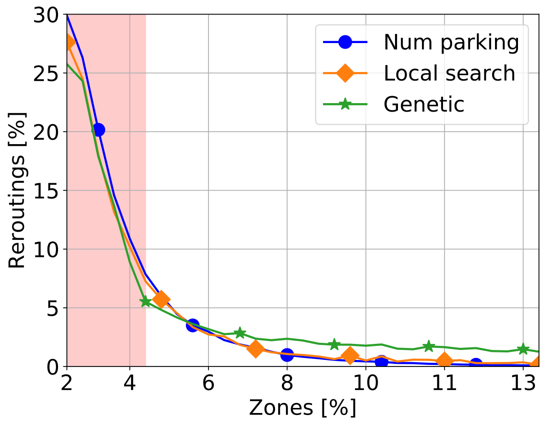
Figure 7.3 reports the two target metrics, for all the optimised configurations. Firstly, in figure 7.3a I compare the infeasible trip percentage. *Num Parking* solution (blue line) has already good performance, reaching 0 with 4.2% of the equipped zones. The *Local Search* (orange line) and the *Genetic* (green line) algorithms are able to further reduce the minimum percentage of zone to equip to guarantee no infeasible trips: 3.8% by the *Local Search*, and to 3.5% with the *Genetic* algorithm, $Z = 10$ and $Z = 9$ zones respectively.



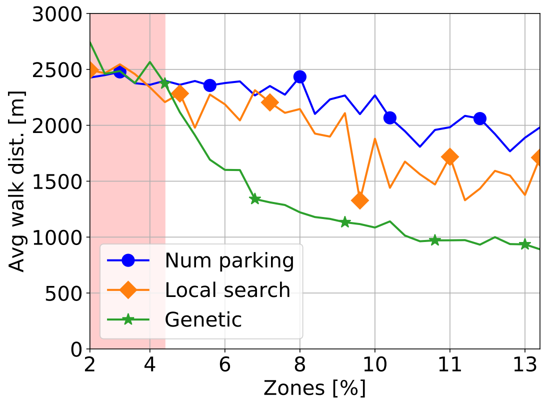
(a) Average state of charge.



(b) Average state of charge.



(c) Rerouted trips percentage.



(d) Rerouted trips percentage.

Figure 7.4: *Genetic* and *Local Search* optimisation results for metrics of interests (*Hybrid* return policy adopted).

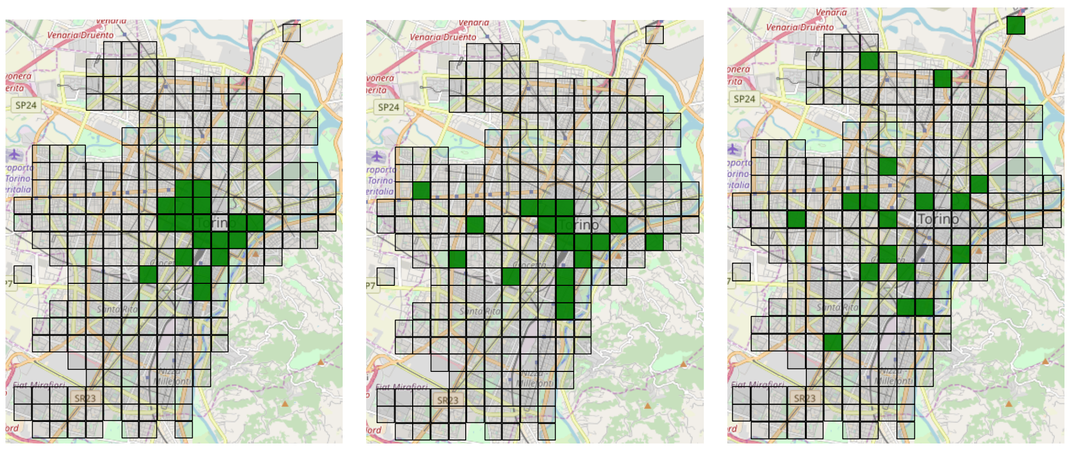
Figure 7.3b reports the walked distance. Focusing in the feasible region, the *Genetic* algorithm confirms the best performance, reducing the distance from more than 200 m to 136 m when 4.2% of the zones are equipped with charging stations and reaching just 30 m at 13%.

In figure 7.4 I further study the new solutions on other metrics. Figure 7.4a reports the percentage of trips ending in a charging station. The more charges are performed, the more time the customer has to spend time plugging/unplugging the car. By minimizing the walked distance, I also reduce this metric since a trip ending with a charge corresponds to 150 m of penalty. Here, the *Local Search* follows the same trend as the *Num parking* with a strong rise. The *Genetic* algorithm shows much better results, from 9% to 14% of trips ending with a charge – half of those found with other solutions. This improvement highlights the better placement of the charging stations. Focus now on figure 7.4b, which reports the average state of charge of the car battery. No major differences are observed here, with all curves almost overlapping up to 7% of the zones.

In a nutshell, the solution found by the *Genetic* algorithm lets customers charge much less frequently while keeping the average state of charge very similar. Consider next figure 7.4c, which details the percentage of reroutes. It is possible to see how the trend is the opposite with respect to the previous ones. Here, the *Genetic* optimised solution show a little higher re-routing percentage than the *Num parking*. In particular, the *Genetic* algorithm reaches 1.3% of reroutes, while *Num parking* decreases down to 0.2% of reroutes. Indeed, the *Genetic* algorithm places charging stations not only where most rentals end, but also so to decrease the customers' average walked distance, i.e., in those places where likely cars are not so frequently parked but that can be quickly reached in case of rerouting. To understand the importance of this difference, in figure 7.4d I evaluate the walking distance a customer has to walk because she suffered a reroute. The *Genetic* algorithm is able to push the walked distance below 1 km, while the *Local Search* generates marginal improvements. In a nutshell, despite customers are rerouted more frequently, on average, they walk for a shorter, and more bearable, distance.

In conclusion, a smart placement of the charging stations is better under different perspectives. The *Genetic* solution, tailored on the data of the usage behaviour, allows us to improve both the system performance, and customers' discomfort, in particular by greatly reducing the number of times they have to charge, and the distance they have to walk.

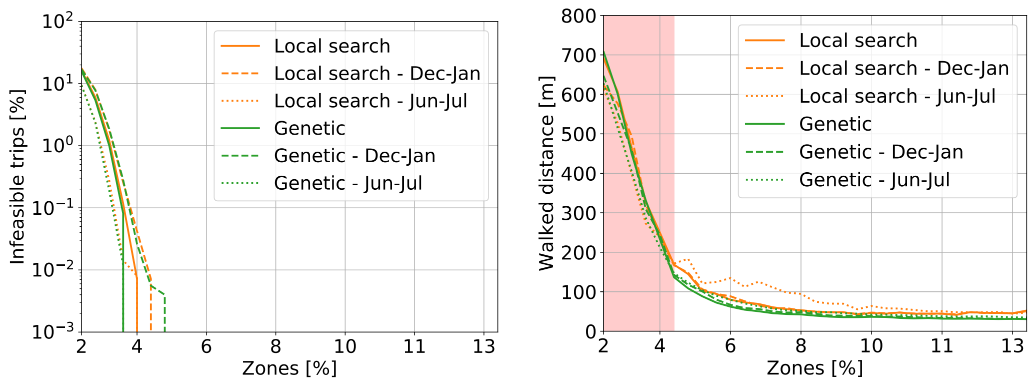
7.6.1 Charging station placement visualisation



(a) Number of Parkings. (b) *Local Search.* (c) *Genetic* algorithm.

Figure 7.5: Different placement of 18 zones (7% of the total) for (a) Number of parkings per zone, (b) *Local Search* and (c) *Genetic* solution. Darker areas have larger values.

To give a feeling about the differences in the solutions found by different algorithms,



(a) Percentage of infeasible trips. Y-Axis is logarithmic. (b) Walked distance, averaged over all trips.

Figure 7.6: Performance of the optimised configuration, tested on other 2 months long data-sets.

figure 7.5 reports the solutions obtained with 7% of the zones equipped with charging stations (i.e., $Z = 18$ zones).

Num parking solution, figure 7.5a, places most of the charging stations in downtown area and near the main train stations. *Local Search*, figure 7.5b, still has many zones in common with *Num parking*, the solution it started from. It just spreads some charging station to cover also some remote zones. The *Genetic* algorithm, figure 7.5c, shows very few zones in common with *Num parking*. Charging stations are spread all over the city, still with more density in the city centre.

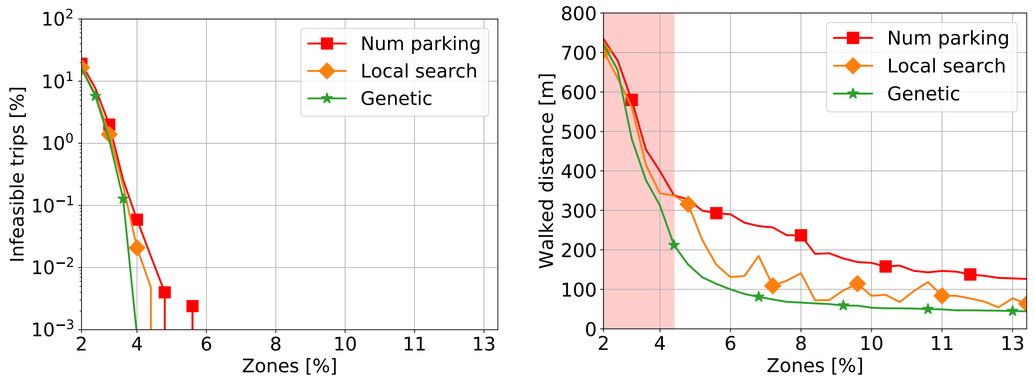
7.6.2 Validation of optimised configurations

The optimised solutions presented in the previous section are built through data-driven simulations. Hence they might over-fit the data of the specific considered period and not be robust to customers' habit changes. To validate those findings, I test the output placement configurations by using independent test traces, different from the one used to run the optimisation. I rely on two traces collected in Turin in two different periods of the year: one in summer, from June to July 2017; the other in winter, from December 2017 to January 2018. I focus on these two periods since in summer and near Christmas holidays the users may exhibit different habits (e.g., customers may rent the cars to go to parks and swimming pools during summer). These anomalies may represent a challenge for the optimised configurations. In the summer trace I record about 100 000 rentals, while in the winter one 128 000 rentals (respectively 8% less and 3% more with respect to the September/October trace). I compute the best station placement considering the September and October 2017 trace, and test system performance using the summer and winter traces.

Figure 7.6 compares results. I consider both *Local Search* and *Genetic* algorithms.

In almost all cases, differences are negligible, showing that the solution is robust. For example, for 13% of zones and considering *Genetic* algorithm, the walked distance on the tests are just 2 m above those in the trace used for optimisation. Notice how in June and July the *Local Search* behaves worse than other cases: this is possibly due to the different mobility patterns in summer, while the Local Search solution could still be too related to the number of parkings in September-October. On the other hand, the solution found by the global genetic algorithm is robust.

7.7 Placement optimisation for *Needed* return policy

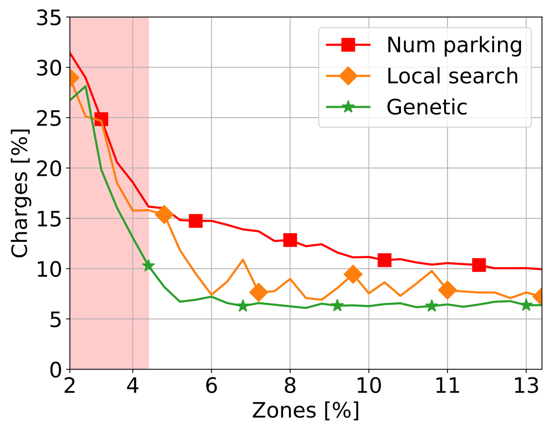


(a) Percentage of infeasible trips. Y-Axis is logarithmic. (b) Walked distance, averaged over all trips.

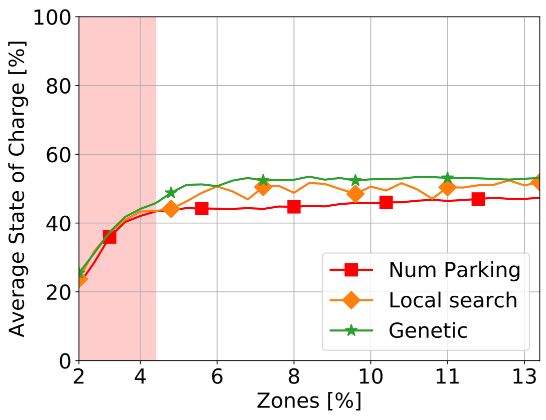
Figure 7.7: Objective metrics to minimise in the optimisation - with *Needed* return policy

Here I briefly report the results for the optimisation experiments of the *Needed* policy. I followed the same procedure explained in Sec. 7.6 for the *Hybrid* return policy. As in that case, the genetic algorithm is able to largely optimise the solution, as reported in figure 7.7, with *local searches* stuck in local minima. In particular, for the walked distance (figure 7.7b), the genetic algorithm is able lower it from 136 m to 45 m at 13% of zones. Still, it doesn't reach the performance of the *Hybrid* policy, i.e., 30 m at 13% of zones.

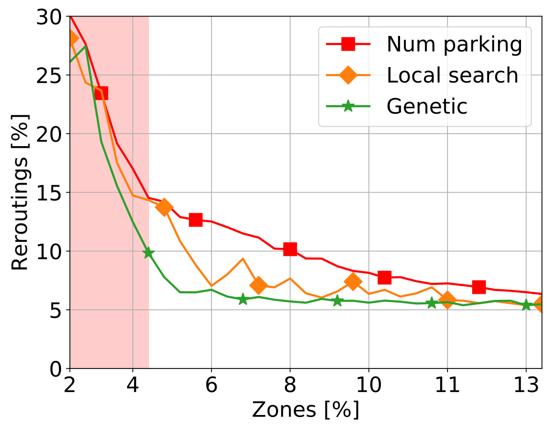
Figure 7.8 reports the other user discomfort metrics. The two optimisation algorithms reduce the charge events (Fig 7.8a). Since the car are recharged less frequently with respect to the *Hybrid* policy it is interesting to evaluate the impact on the *Average Stage of Charge*. figure 7.8b reports this metric for the different placement algorithms. As we expected, the *Average Stage of Charge* is lower with respect to the *Hybrid policy*. Interestingly, for both optimized solutions this value saturate almost immediately at 55% in the feasible region. However, the average state of charge (figure 7.8b) is always higher in the optimised solutions than in the *Num parking* configuration. This



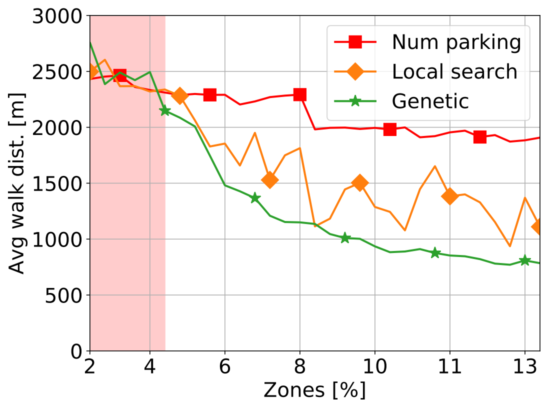
(a) Charges percentage.



(b) Average state of charge.



(c) Rerouted trips percentage.



(d) Walked distance when rerouted.

Figure 7.8: *Genetic* and *local search* optimization results for metrics of interests (*Needed* return policy adopted).

further demonstrates that a smarter placement allows the car to get more energy for each charge. Figure 7.8c reports the re-routing percentage for the different algorithms. As expected, the values are larger than with the *Hybrid* policy, since no opportunistic charge is performed. This different is particularly evident when a few charging stations are present e.g., with 5% of charging stations the rerouting are 14% for the *Num parking* heuristic, 14% for the *Local search*, and 6% for the *Genetic* algorithm. However, the genetic algorithm is able to quickly reach a small value of re-routings, hence better exploiting every charge possibility. Finally, I analyse the distance a user has to walk when rerouted (Figure 7.8d). Here, with respect to the *Hybrid* policy case, the *local search* shows larger deviations from the *Num parking*. The genetic algorithm reaches values of about 800 m, even below the ones reached with the *Hybrid* policy.

In conclusion, within this range of zones equipped with charging stations, a smart placement with the *Needed* policy approaches the *Hybrid* policy results.

7.8 Discussion and Implication

The mobility model I use for optimising the placement of charging station reflects customers habits. However, there are other aspects of car sharing system design that one could consider. In the following, I briefly discuss two of them.

7.8.1 Scalability

All the analyses I performed heavily rely on the events trace derived real rentals in the analysed city. By using this data I use simulations to study placement solutions which cope with the current amount of traffic and usage of the FFCS. While in a short term this solution is optimal, there is no guarantee whether it will be valid in the future in case of a strong increase in the car sharing usage, e.g., when popularity increases by orders of magnitude. To tackle this problem, it is possible to still leverage rental data to infer a model about car sharing usage patterns in time and space, where the demand can be easily controlled by increasing the frequency of rentals. This model can then be used to create synthetic traces with an increasing car sharing demand, and use simulations to assess overall system performance. As such, the methodology I designed is generic and could be used to study different *What-If* cases.

Directly linked to the scalability problem, another important aspect is the possibility to add new charging stations when car sharing demand changes. By analysing the data collected, this methodology can be used to consider the greedy placement of new charging stations on the top of those already present.

7.8.2 Economical Aspects

Economical aspects play too a key role in the placement decision process. In this work, I decided to study only the feasibility of the EV based FFCS system design by considering as few charging stations as possible, leaving for a next step the detailed cost estimation. The cost of the infrastructure creation and management can largely vary depending on different variables such as country, city, incentives. Related to this first class of costs, authors in [73] give a first estimation of installation cost, which can be of up to 5 500 USD per pole in the USA. While analysing these costs and developing a business plan, it is also important to evaluate which parties could be interested on running a business around it, e.g., either the municipality or a third party company could offer the infrastructure as a service to FFCS providers and other customers with electric vehicles.

The second group of variables consider the earnings models and the variable costs of the FFCS provider, like energy and cars. This kind of data requires a careful analysis to get a reliable estimation. Authors in [66] suppose a marginal profit of 75% of the fare considering a FFCS provider in the city of Vancouver. However, to correctly estimate the net profit several aspects need to be considered, such as the fee per minute, the actual duration of the rental costs and incentives for reroutes.

7.9 Conclusions

Designing an electric free floating car sharing systems leads to many interesting problems and trade-off between usability, costs and benefits for the customers. In this work, I built on actual rental traces to study via accurate simulations the impact of different charging stations placement and charging policies. I selected Turin as a case study and using 2 months of rentals recorded from a currently operational FFCS that I use to run trace driven simulations.

I have shown that few charging stations are enough to make the system self-sustainable. Important is the customers collaborations, so that they voluntarily returns to the cars to charging stations when available This data driven results show that just 5% of the city zones that are equipped with charging stations (13 in total, 52 poles) make all trips feasible with an electric car fleet. Moreover, through a charging station placement based on a genetic optimisation algorithm, it is possible to minimise the discomfort for the customers that would be (rarely) asked to bring the car for charging. For example, with 18 charging stations (72 poles in total), on average a customer would walk only 40 m to reach its desired destination. While these numbers will change in different cities, the data driven approach I propose naturally fits the global optimisation algorithm that is able to optimise placement while considering complex customers habits.

I leave for future work the simulations of scenarios with new technologies, such as deployment of faster charging poles and larger batteries, and the scalability in terms of number of customers and fleet size.

Chapter 8

FFCS Usage Prediction with Open Socio-Demographic Data

This work is mainly extracted from my paper *On Car-Sharing Usage Prediction with Open Socio-Demographic Data*, published on the journal Electronics on January 2020 ([74]). The entire work was carried in collaboration with the Universidade Federal do Minas Gerais, Belo Horizonte, Brazil. My contribution are mainly related data collection, data augmenting and spatial analyses in sections 8.3, 8.4 and 8.6.

8.1 Introduction

Transportation in urban areas is among the top challenges to improve people's quality of life and to reduce pollution. Historically, private vehicles have been the preferred mode of transportation. Orthogonally, governments invest in public transportation systems to offer alternatives to reduce traffic and pollution. With the rise of the sharing economy, we are now witnessing a transition towards new forms of shared mobility, which have spurred the interest of both the research community and the private companies.

Car sharing is an evolution of the classic car rental model. Here, users can rent cars on-demand for a short period, e.g., a twenty-minute trip across town. In particular, Free Floating Car Sharing (FFCS) services allow customers to rent and return the cars everywhere inside a operative area in a city. Customers book, unlock and return the car by using an application on their smartphones. One such service is Car2go [1], which currently operates in several cities around the world. In the FFCS implementation, of which Car2go is an example, the provider bills the user only for the time spent driving, with simple minute-based fares which factors all costs. Some studies demonstrate that

[1]https://www.car2go.com/

a massive adoption of car sharing service can improve mobility as well as reduce costs and pollution [75, 27, 55].

To properly design and manage a FFCS service, a provider needs to know the demand for cars over different periods of the day, and over the different areas of the city. The prediction of FFCS demand patterns is thus fundamental for an adequate provisioning of the service. Armed with good predictions, the provider can better plan long term system management, e.g., whether to extend the operative area to those neighborhoods with expected customer growth. Similarly, it can implement short term dynamic relocation policies to better meet the demand in the next hours [76, 43, 46].

In this work we investigate the usage dynamics of a real FFCS service. We aim at assessing how state-of-the-art machine learning algorithms can help FFCS providers and policy makers in predicting the demand, both over time and across different spatial regions. More specifically, we leverage a dataset of real rides from cities where Car2go is offering its FFCS service. We consider as a case study the city of Vancouver, Canada, the city with the highest demand for cars in our dataset. We rely on more than 1 million rentals covering 9 months in 2017 [17]. We augment the dataset by exploiting a rich and heterogeneous open dataset, namely the 2016 Vancouver Municipality census.[2] This second dataset comprises more than 800 features, which detail very diverse information about shops in each neighborhood, weather conditions, residents, rate of emergency calls throughout the day, etc. The goal is to first assess to which extent it is possible to predict the FFCS demand over time and space, and second, which of the features have a higher prediction power.

The work focuses on two scenarios. In the first scenario, we investigate how to predict the demand for cars in the future considering past usage. This is fundamental for managing the FFCS fleet both in the short term (e.g., implementing relocation policies during service peak time), and in the long-term (e.g., to properly match the fleet size to the future system growth). To this end, we analyse machine learning algorithms that are considered state of art, from simple Linear Regression and traditional Seasonal Auto Regressive Integrated Moving Average (SARIMA) models, to Random Forests Regression (RFR), Support Vector Regression (SVR) and latest approaches based on Long Short-Term Memory Neural Network (NN) [77, 78]. With the increasing complexity of these models, we aim at assessing not only how they perform in our target prediction task, but also to which extent one would need to embrace a complex model (such as NNs are) or rather simpler and more informative models (like linear regression and RFR are).

In the second scenario, I correlate socio-demographic indicators with FFCS demand. I predict the demand of cars in a neighborhood without past data, using only socio-demographic data. This problem is often referred to as a green field or cold start approach. In this case, the operator is interested in knowing what is the expected system usage in a new neighborhood (or even a new city) based only on socio-demographic

[2]https://opendata.vancouver.ca/pages/home/

data. We map the FFCS demand to Vancouver neighborhoods, and associate them to the socio-demographic data coming from the official Vancouver census. I then use machine learning techniques to highlight the relationship between demographics and customers' mobility.

We aim at answering the following research questions: i) Using modern machine learning methodologies, and armed with a rich socio-demographic data, would one be able to predict the temporal mobility patterns in a city? And ii) which would be the most important socio-demographic data to use for this task?

Through a series of experiments, we show that the temporal prediction of rentals can be solved with errors as low as 10%. Interestingly, Random Forest Regression turns out to perform stably better than the other models, including Neural Networks, for this task. When considering the mobility prediction using only socio-demographic data, we obtain errors in the 40-50% range. While this performance may not be accurate enough for a precise planning, this prediction still would be useful for operators willing to decide, e.g., to which new areas of the city to extend their service. Interestingly, our models allow us also to observe what features are the most useful for the prediction problem, a precious information for providers and regulators that wish understand FFCS systems – to decide, for instance, in which new cities to start a service (green field problem). This work suggests, for example, that the density of people commuting by walk and the number of emergency calls in a neighborhood are important factors for predicting the number of rentals that will start there. We note that emergency calls are used as a proxy for human activity, i.e., the more human activity the larger the number of emergency calls. Given this assumption, we can leverage the information about the volume of emergency calls to improve prediction at different time of the day. AS for the temporal prediction, knowing the weather conditions in the near future would improve prediction too.

After overviewing the related work in Section 8.2, we describe the data collection methodology we adopt in Section 8.3. Section 8.4 provides a characterization of the datasets, while Section 8.5 and Section 8.6 provide details about the methodologies and results for the temporal and spatial prediction, respectively. Finally, Section 8.7 summarizes our findings.

8.2 Related work

With the easiness of collecting data and the ability to build and train off-the-shelf machine learning solutions, researchers have started applying data driven approaches in the context of transportation. Previous work [79] addressed traffic modelling and prediction with real traffic data, and proposes strategies to improve congestion prediction using Kalman filters, showing how traffic is stationary in time. Other studies [80] proposed new approaches, based on a multivariate extension of non-parametric regression, to predict traffic patterns, with the goal of counteracting traffic congestion. While similar in spirit, this work focuses on FFCS services explicitly, and uses a much richer dataset as well as more advanced machine learning algorithms.

Focusing on car sharing, early work focused on estimating demand using activity-based micro-simulation to model how agents move around in a city [25]. Later on, as data from operative car sharing platforms became available, researchers started using real data to analyze mobility demand. For instance, previous work [81, 27] proposed a demand model to forecast the modal split of the urban transport demand. Similarly, other studies [24] investigated the Mobility-as-a-Service market, where FFCS is one of the implementations, and pointed out how FFCS supply can push the users to avoid purchasing a new car, which would lead to a reduction of CO_2 emission. Yet, none of these prior studies focused on car sharing demand prediction.

Along the same lines, other studies [35] made a large survey covering a Swiss station-based car sharing service. The results confirmed that FFCS is preferred as a fast alternative to public transportation and the subscription depends on the car sharing implementation (business model). Previous work [76] also proposed a simple binary logistic model for predicting car sharing subscribers in Switzerland, considering the relationship between potential membership and service availability. This relationship was then used to identify areas with unmet demand, that is, areas where new car sharing stations could be placed.

Other studies [82, 22] conducted a detailed characterization of a car sharing system in Munich and Berlin. Similarly to this work, they identified features correlated with the demand for shared cars in the target cities. However, this work differs from their in the sense that we here analyze a much larger set of features, including demographics and economic data, and consider multiple prediction models. We focus on demand prediction, facing both time and space dimensions, and provide a thorough comparison and guidelines for future directions.

In this previous work [31], we analyzed in depth the usage of different car sharing systems in Vancouver. Based on this data, we developed a model of FFCS usage and built a simulator to design new systems based on electric vehicles [43]. In particular, we tackled the charging station placement problem, showing that the optimal placement requires few stations to satisfy charging requests in different cities [46].

To the best of our knowledge, we are the first to face the demand prediction problem in Free Floating Car Sharing Systems tackling both the temporal and spatial prediction with a real world heterogeneous dataset. The demand prediction problem (or its variations) has been tackled in other domains [83, 84], we here focus on multiple prediction tasks (long-term, short-term) accross different aspects (temporal and spatial) on the car sharing domain.

Furthermore, while previous work [85] focused on the temporal prediction of car sharing demand in a very short-term basis (demand prediction in the next few minutes), in this work we focus on the problem at different time scales. We also compare several prediction strategies and analyze how the temporal prediction problem relates to the spatial prediction one. Moreover, we are the first to use a very heterogeneous dataset including dozens of features to tackle the prediction problems. This allows us to provide insights on which of those features are the most important ones to solve the prediction problems as well as to have a broader perspective on the challenges involved in car sharing prediction.

8.3 Data gathering methodology

8.3.1 FFCS data collection and Socio-demographic, weather and other open data

In this work we used real car2go users' data, a popular FFCS system that offers its services in more than 25 cities and 3.6 million customers in 2019. A detailed description of data collection is reported in chapter 2. In particular we focused our considerations in Vancouver. A deep characterization is depicted chapter 4, but some brief key points are reported later in this chapter.

In addition to information about rentals in the city of Vancouver, we also use socio-demographic data as input to car usage prediction algorithms. Specifically, we consider the Vancouver census open data, which divides the city in 22 official neighborhoods[3]. This work uses this same spatial division. For each neighborhood, the census dataset provides detailed socio-demographic information such as number of residents in a given age range, their income, household compositions, and commuting habits. The census also reports information about services that are located in the neighborhoods, e.g., shops, bus stops, and parking places. In total, the census presents more than 800 socio-demographic and other spatial features. Among those, we manually selected 83 features that might be related to human mobility.[4] In addition, we also consider i) the distance to downtown – computed as the distance from the neighborhood to the downtown neighborhood (considered as the central area);[5] ii) an indicator of human activity, measured by the number of emergency calls per time bin (obtained from the Vancouver census); and iii) the hourly weather for Vancouver – as directly available from the OpenWeather project.[6] For each of the 22 neighborhoods, we normalize each numerical feature by the area of the neighborhood. Our goal is to include a superset of features possibly correlated with human mobility and thus car rental prediction, so as to provide the machine learning algorithms with an input dataset as rich and diverse as possible to learn from.

[3]https://opendata.vancouver.ca/pages/home/

[4]The list of features is available at https://opendata.vancouver.ca/pages/census-local-area-profiles-2016-attributes/

[5]We use the neighborhoods central points for distance computation.

[6]https://openweathermap.org/history-bulk

8.4 Dataset overview

In this section I first provide an overview of the data at our disposal offering insights into the diversity and heterogeneity present both in the temporal and spatial FFCS usage patterns as well as in the socio-demographic data.

8.4.1 FFCS temporal characterization

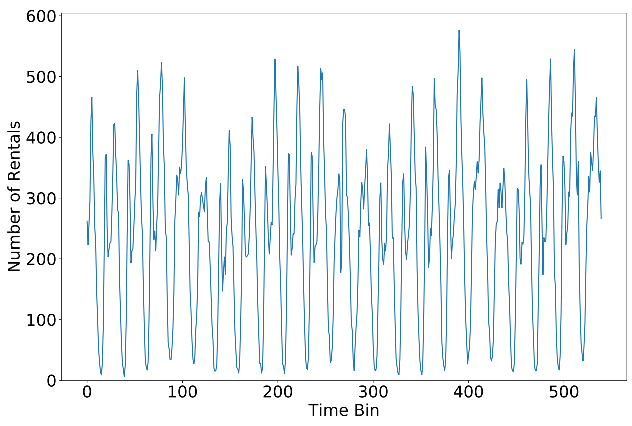
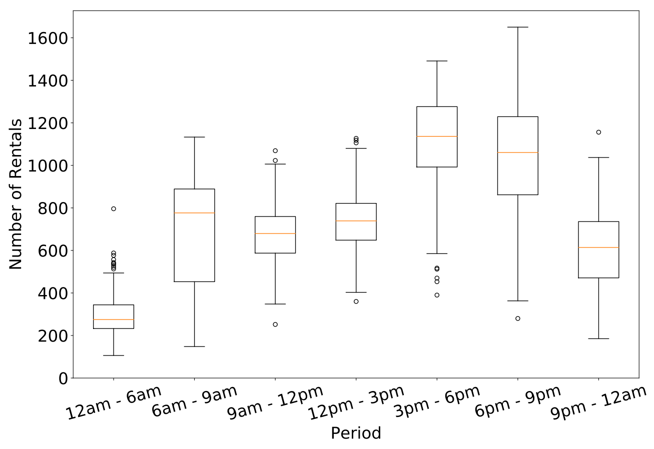


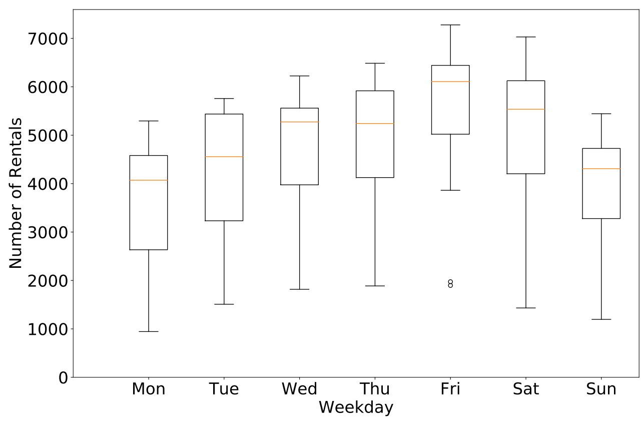
Figure 8.1: Time series of starting rentals in September 2017 aggregated per hour. The difference in the total number of time bins and the actual number of hours in the month are due to missing data (crawler failures).

I start by showing the temporal evolution of rentals over time. Figure 8.1 shows the total number of starting rentals per hour in the whole city during part of September 2017. Even if it is possible to spot some periodicity, there is a lot of variability that makes the prediction problem not straightforward. For the analyses, from now on we aggregate rentals both in time and in space. Specifically, given a neighborhood we consider the fraction of rentals *starting* and *ending* there. We aggregate the time series of rentals into 7 time bins per each day, namely from midnight to 6am (night period), and then every 3 hours. This time granularity is typically used for system design and control [86]. The rationale is to provide the FFCS company that actionable information on the demand for cars, e.g., to schedule car maintenance or implement relocation policies. A one-hour period is often too short for the company to be able to respond to changes in demand.

To give more details about the variability of the data, figure 8.2a shows boxplots of the numbers of rentals starting in each time bin. Each boxplot represents the quartiles



(a) Boxplots of number of rentals starting in each time bin



(b) Boxplots of number of rentals starting in each day of the week

Figure 8.2: Temporal characterization of number of rentals. Boxplots highlighting the variability over the day for the same time bin of the day (top plots), and over different days (bottom plots).

of the distribution, with outliers shown as points.[7] The series shows large variability, with peaks during early mornings (6am-9am) and afternoon (3pm-6pm and 6pm-9pm),

[7]We consider as outliers measures that are outside the mean ± 2.698 times the standard deviation range.

and with low values during nighttime (12am-6am). Figure 8.2b shows boxplots of the total number of rentals grouped per day of the week. The number of rentals peaks on Fridays, with significantly lower values registered on Sundays and Mondays. Again, we observe a quite sizeable variability over the days, as observed by in the sizes of the boxplots. Such variability in the number of rentals hints at the fact that prediction models have to be able to deal with sizeable temporal variations in the demand for cars.

8.4.2 FFCS spatial characterization

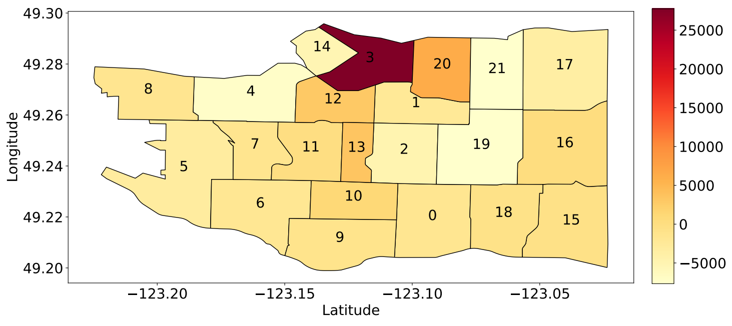
I now provide an insight how these numbers vary across different areas of the city. Rather than providing a complete characterization of the origin/destination matrix (which is outside the scope of this work), I here focus on particular examples to showcase the spatial variability in the demand for cars. I focus on the morning and afternoon peak time bins (6am-9am and 6pm-9pm). For each neighborhood I compute the *net flow* defined as the difference between the number of rentals starting from that neighborhood, and the number of rentals arriving at that neighborhood during the specified time period. I consider the cumulative net flow in September 2017. Figure 8.3 depicts the results with a heat map. Darker red neighborhoods mean that arrivals exceed departures, i.e., the neighborhood is attracting vehicles. Conversely, lighter colors imply that more vehicles are departing from that neighborhood than arriving in it. Numbers identify different neighborhoods. The downtown business area (number 3) attracts a lot of rides in the morning period (figure 8.3a), while the opposite pattern is seen during the afternoon period (figure 8.3b). In general, it is possible to assert that the FFCS demand is higher in the peak hours, and the cars flow towards downtown in the morning and towards residential areas in the afternoon. This is clearly visible in figure 8.4 which reports the net flow for two neighborhoods for each hour of the day, namely the downtown neighborhood (number 3), and the Grandview-Woodland (number 21) neighborhood, a residential area close to downtown.

8.4.3 Socio-demographic and weather data characterisation

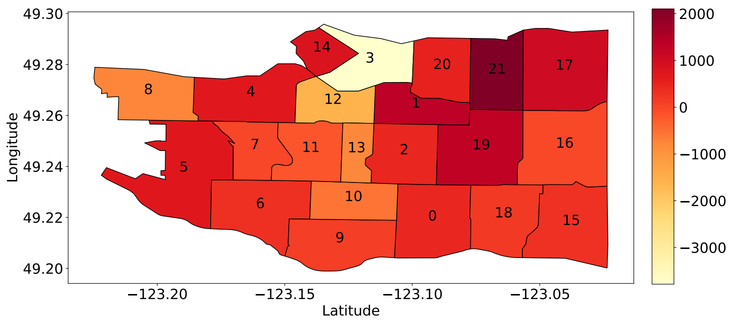
I now provide some examples of the socio-demographic and open data. Figure 8.5 reports the weather condition during the month of September 2017. Being it a categorical variable, I assign to each weather condition a different value on the y-axis. As expected, the weather conditions change over time quite frequently. Moreover, no visible correlation is found when comparing the weather conditions with the number of rentals in figure 8.1.

Similarly, figures 8.6a and 8.6b show the number of high-income households and the number of emergency calls per day for each neighborhood, respectively. Also in this case, it is hard to see any clear correlation with the net flow per neighborhood reported in figures 8.3. The scenario is similar considering other socio-demographic features.

Despite the non-linear correlations between the socio-demographic data and rentals,



(a) 6am - 9am rental net flow



(b) 6pm - 9pm rental net flow

Figure 8.3: Heatmap of net flow for each neighborhood in Vancouver. The more the area is red, the higher are the arrivals with respect to the departures. Neighborhood numbering is shown (from 0 to 21)

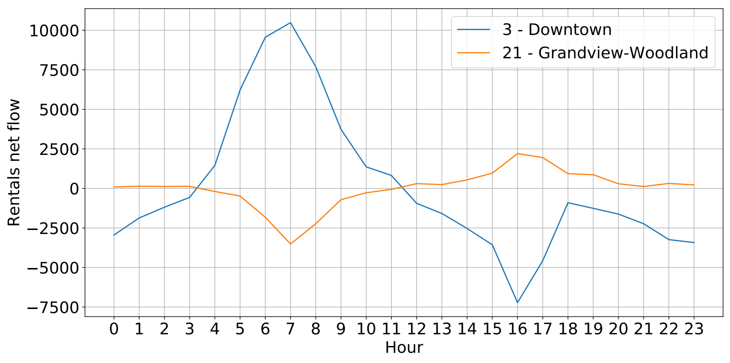


Figure 8.4: Total net flow in September 2017 for Downtown (neighborhood 3) and Grandview-Woodland (neighborhood 21) over different hours of the day

it is possible that the combination of multiple features help the prediction of car rentals, as we will discuss in the next sections. This is exactly what the machine learning algorithms aim at, i.e., building a model from data, leveraging correlation from multiple variables that, considered together, carry enough information to predict system usage. Thus, we let the machine learning model decide if and how to factor different features in the prediction model.

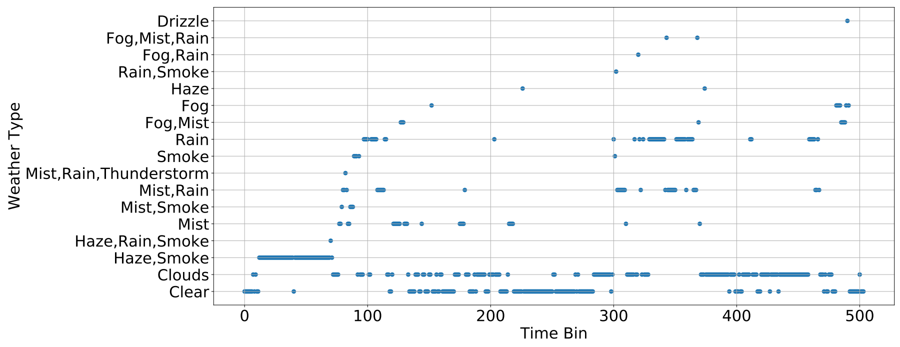


Figure 8.5: Time series of weather conditions per hour during September 2017. Each point in the plot represents an occurred weather type.

8.5 Temporal predictions of rentals

In this section, we describe the task of predicting the number of rentals in the whole city at a given time in the future. Eventually, the same methodology could be applied for each neighborhood. This prediction can exploit historical data, i.e., given the time series of rentals in the past, predict the number of rentals in the future. If only the past time series are used, the problem falls in the univariate regression class, i.e., the prediction is based only on past data of the same target variable. Let $x(t)$ be the target variable, i.e., the number of rentals at time t. In the case of prediction with historical data, we predict

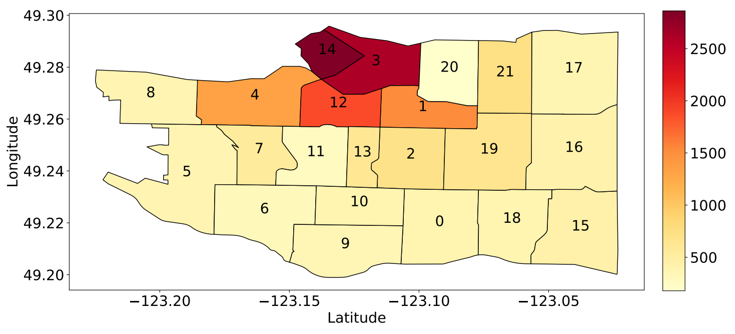
$$x(t+j) = f(x(t), x(t-1), \dots, x(t-k)),\ j > 0, \tag{8.1}$$

as a function $f()$ of the past $k+1$ data points of x itself. j is the horizon of the prediction.

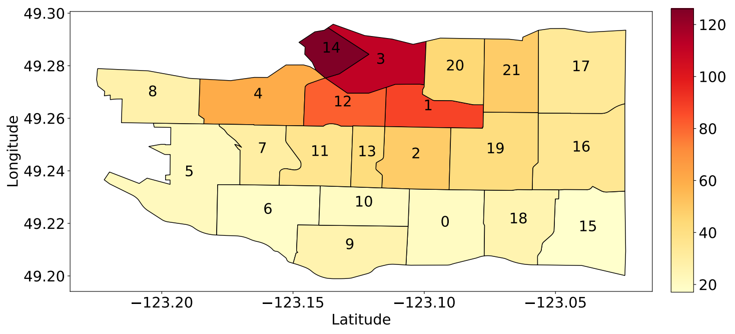
if other information are present, it possible to predicict a more generic model to consider the dependence to other variables The goal is to predict

$$x(t+j) = g(y_1, y_2, \dots, y_l),\ j > 0, \tag{8.2}$$

where $\{y_i\}$ are different variables – possibly other time series themselves (including x) – and g is the model that allows to predict x at time $t+j$. This problem is a multivariate regression problem, where multiple features are used to predict the target variable x.



(a) Maximal income range density per neighborhood



(b) Emergency calls per day in each neighborhood

Figure 8.6: Heatmap of a sample of demographic (top) and socio-demographic (bottom) data at our disposals. These two samples look quite correlated.

Considering the time horizon of the prediction, it is possible to formulate two versions of the problem: predict the long-term or short-term usage. In the first case, we build and train a single model using all available data to predict the system usage in the next months. In the short-term version, we target the prediction of the next time bin $t + 1$ only, i.e., $j = 1$. In this second case, we build and update a new model at each time bin by adding the latest recorded number of rentals to the training set as soon as it becomes available.

Both predictions are important for the car sharing provider. For instance, the long-term predictions are important for the company to know if their fleet size is enough to keep up with the expected demand. The short-term is important for the company to know when to take a car down for maintenance, or when and where cars should be eventually relocated to those neighborhoods where the demand is expected to increase shortly. While for long-term prediction we use the time series of the rentals and information about day of the week and hour of the day, for short prediction we can use also the near future weather condition information.

This work, we consider discrete time, i.e., we split time into fixed size time intervals as defined in the aggregation step – see Section 8.4. We then build and train several machine learning models to tackle each aforementioned problem. The goal is to compare algorithms in terms of accuracy of the prediction and complexity of the model. At last, we are also interested in considering models that are interpretable, i.e., that allow us to understand which are the most important features that affect car sharing usage in large cities. We evaluate all models considering three metrics: APE (absolute percentage error), MAPE (mean absolute percentage error), and RMSE (root mean square error) over the validation set. The APE is defined as

$$APE = 100 \sum_{t_i \in V} \frac{\mid x(t_i) - \hat{x}(t_i) \mid}{x(t_i)}, \tag{8.3}$$

where V is the validation set, $x(t_i)$ is the actual value of the data at moment t_i and $\hat{x}(t_i)$ is the predicted value. The MAPE is then given by

$$MAPE = \frac{1}{|V|} \times APE. \tag{8.4}$$

And the RMSE is defined as

$$RMSE = \sqrt{\frac{1}{|V|} \sum_{t_i \in V} \Big(x(t_i) - \hat{x}(t_i) \Big)^2}. \tag{8.5}$$

8.5.1 Prediction models

We use off-the-shelf machine learning models both for the long-term and short-term scenarios. We evaluate univariate models: a simple baseline (BL) approach, the autoregressive moving average (ARIMA) and the seasonal autoregressive moving average (SARIMA) algorithms. Univariate models do not account for the influence of other time-variant factors such as weather conditions, time of day, number of emergency calls, etc. To account for that, we also investigate the performance of linear regression, random forests Regression (RFR), Support Vector Regression (SVR), and long-term short-term memory neural networks (NN).

We add categorical features (the day of the week and weather, for instance) to these algorithms in order to improve on the univariate models. Following correct practices [87], we represent each categorical feature as many binary variables, one for each category. For example, when representing a given weather type, the corresponding binary variable will be set to *True* while all the other weather-related variables to *False*. We used the algorithms implementation in Python libraries `scikit-learn`[8] [88] and Keras[9]. For details about each model, we refer the reader to [78]. In this implementations, we

[8]https://scikit-learn.org/

[9]https://keras.io/

started with the library's default hyperparameters and conducted a grid search in order to find a set of such parameters that worked well with the described models. We report the range of the grid search along with the description of the models below.

Baseline. A simple approach to determine $x(t+j)$ in a time bin is to take the average number of rentals in the same time bins in the available past days. We compare all the prediction models to this baseline.

ARIMA. ARIMA (autoregressive integrated moving average) is widely used to predict time series data. ARIMA models are a combination of autoregressive models with moving average models. The creation of an ARIMA model involves specifying three parameters (p, d, q). The d parameter measures how many times we have to differentiate the data to obtain stationary data. After determining d, we use sample partial auto correlation function to get the value p. Finally, we determine the order q by looking at the sample auto correlation function of the differentiated data. For simplicity, we restricted the grid search to find the best parameters values to the range $[0, 3]$. The combination that gave the best results is $(p, d, q) = (2, 0, 1)$.

SARIMA. A SARIMA model incorporates the seasonality (periodicity) of the data into an ARIMA model, enhancing its predictive power. For instance, when modeling a time series, it is often the case that the data has a daily, weekly, or monthly periodicity. We used previous ARIMA model with an additional explicit daily seasonal component ($p = 7$ as the number of time bins in a day in this case).

Linear Regression. We fit a linear model, by finding the coefficients that multiply each feature.

SVR. In the experiments, we use a Support Vector Regression (SVR) model with the following combination of parameters, which produced the best results among the values we tested: $C = 1000$, $\gamma = 0.1$, and $\epsilon = 0.1$, with the RBF kernel. The values for the parameters $\gamma = 0.1$, and $\epsilon = 0.1$ were evaluated in the range $[0, 1]$, and for the C parameter we considered the range $[1, 10000]$, using exponential steps. The value 1000 was chosen once it provided a reasonable balance between model performance and generality.

RFR. Random Forest Regression is an ensemble learning method that can be used for regression. The decision is based on the outcome of many decision trees, each of which is built with a random subset of the features. One advantage of random forests over linear regression is that the forest model is able to capture the non-linearity. Another advantage of RFR is that they are interpretable models, i.e. they offer a ranking of the most important features for the prediction problem. Here, we use 50 decision trees[10]. In this model, we used the default library parameters, but we evaluated it with different numbers of trees, for which the results are shown in the next sections.

Neural Networks. We also consider a Long Short-Term Memory (LSTM) Neural

[10]Here, interpretable refers to the fact that it is possible to understand the decision taken by the classification model. However, interpretability has not to be confused with explainability, which refers to the motivations of the decision. The latter is only possibly via domain knowledge.

Network model. LSTMs have a memory that helps capturing past trends in the data, which may favor the prediction task. We experimented with several different architectures. The best results were obtained with a three layer architecture where the input layer has 64 neurons (one for each feature), the dense layer has 4 neurons, and the output layer has one neuron. We tested different configurations for the architecture: the number of neurons was varied in the range $[4, 128]$ for the first layer, and in the range $[4, 32]$ for the second layer. Because of the nature of the task (regression and not classification), the number of neurons in the third layer was set to one. In the experiments, to balance prediction accuracy and training time, the model was trained for 50 epochs. As we will see, increasing the number of epochs to more than 50 has no significant effect (less than 1% reduction in the MAPE, on average) on the performance of the model.

8.5.2 Long-term predictions - Results

Here we predict the FFCS demand for cars in the future months given a model built on the previous months. We use in the experiments the nine months of 2017 of car sharing usage of Vancouver. Given the volume of rentals in the training period, we try to predict the number of rentals in the validation period. For that, we use a model that is trained once and then used to perform all the predictions in the validation period. The training set consists in the volume of rentals for the first six months, and the validation data consists of volume of rentals for the next three months.

Table 8.1 shows the average mean absolute percentage error (MAPE), the standard deviation of the APE, and the RMSE for each of the prediction models. The models that rely only on the time series (ARIMA and SARIMA) are able to capture some patterns in the data, as their performance is considerably better than the baseline. However, the multivariate models perform better, with Random Forest Regression reaching the best performances. In figure 8.7 we show the comparison between the actual values and the prediction in one month of the validation set using the Random Forest Regression model (orange dashed line). Overall the model is able to predict quite well the daily and weekly periodicity of rentals, but in general slightly underestimates the actual number of rentals. This could be due to the fact the training period refers to the first six months of the year, during which the average number of rentals is lower than during the validation period in fall.

8.5.3 Short-term predictions - Results

We now tackle the problem of predicting the demand of cars in a city in the next time bin. Differently from the long-term predictions we use adaptive models, hence the model is re-trained every time new data is made available, so then we can add it to the training set. We here focus on the following prediction task: given the volume of rentals per time bin period for a specific number of past days and the weather conditions, predict the number of rentals in the next time bin period.

Prediction Model	MAPE [%]	σ(APE) [%]	RMSE
Baseline	40.05	44.95	321.32
ARIMA	25.53	19.68	238.87
SARIMA	21.15	21.74	159.17
Linear Regression	15.80	15.61	178.57
Support Vector Regression	15.12	16.14	179.99
Random Forest Regression	14.63	11.62	157.40
Neural Networks	15.83	16.60	187.08

Table 8.1: Long-term temporal prediction - Mean Absolute Percentage Error (MAPE), Standard Deviation of the Absolute Percentage Error (APE) and Root Mean Square Error (RMSE) for each prediction model in the validation set.

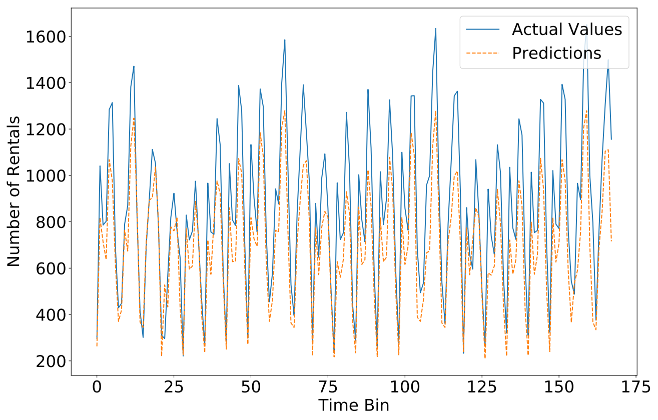


Figure 8.7: Long-term temporal prediction - Performance of the RFR model in one month of the validation set. The difference in the total number of time bins and the actual number of hours in the month are due to missing data (crawler failures).

We study this prediction task using two approaches: expanding window and sliding window. In the *expanding window* approach, after making the first prediction, we add the actual value to the training set, therefore increasing the amount of data available for training in the next step. To train the models, we first set aside 24 days of data for validation, and start with 28 days of training data. In the *sliding window* approach, after making the prediction we remove the oldest training data and add the actual value to the training set. Therefore, the training set size is always the same during the evaluation of the models. To train the models, we consider different sliding windows sizes (from 7 to 28 days), and validate on the same validation set of 24 days as with the expanding window.

	Expanding Window (starting: 28 days)		
Prediction Model	**MAPE [%]**	**σ(APE) [%]**	**RMSE**
Baseline	20.12	16.64	195.53
ARIMA	36.01	35.87	306.80
SARIMA	17.60	20.01	160.42
Linear Regression	18.28	20.38	179.11
Support Vector Regression	12.22	15.62	128.72
Random Forests Regression	9.71	8.34	104.99
Neural Networks	10.52	12.93	128.84
	Sliding Window (starting: 28 days)		
Prediction Model	**MAPE [%]**	**σ(APE) [%]**	**RMSE**
Baseline	20.12	16.64	195.53
ARIMA	36.52	36.60	305.50
SARIMA	18.02	21.75	163.94
Linear Regression	18.11	20.55	178.61
Support Vector Regression	12.87	18.52	136.14
Random Forests Regression	10.08	12.23	109.47
Neural Networks	10.52	12.74	123.55

Table 8.2: Short-term temporal prediction - Mean Absolute Percentage Error (MAPE), Absolute Percentage Error (APE), and Root Mean Squared Error (RMSE), for each prediction model in the validation set.

In Table 8.2, we compare the performance of all models using the two approaches. The best results for the sliding window approach were obtained with the largest possible window (28 days). The expanding window approach offers slightly better results, which can be attributed to the fact that the model can exploit more data and the patterns are not changing rapidly in time. Again, the multivariate models, and in particular the Random Forest Regression model, reach the best performance. Interestingly, the Neural Network model performs similarly to other models, suggesting that, for this specific use case, a simple and more interpretable model like a RFR is enough. Furthermore, as shown in figure 8.8, increasing the number of epochs does not have a significant effect on the performance of the Neural Networks model.

We show in figure 8.9 the performance of the best model, i.e., RFR with expanding window. In this short-term formulation of the problem the prediction naturally adapts to changes over time, obtaining better predictions with respect to long-term prediction. Moreover, the weather data also provides useful information.

We now explore the importance of each feature for the model by analyzing the RFR

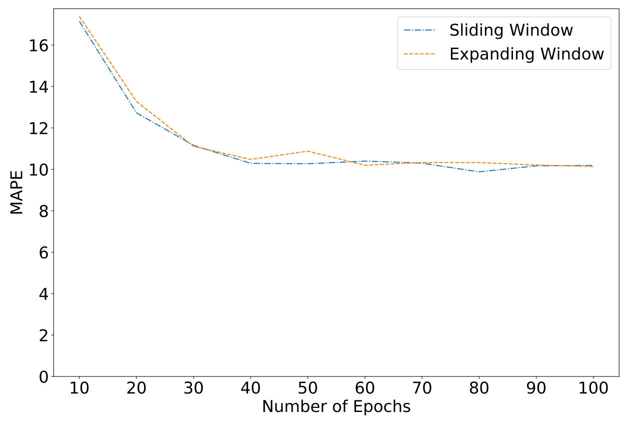


Figure 8.8: Effect of the number of epochs on the performance (MAPE) of the Neural Networks model.

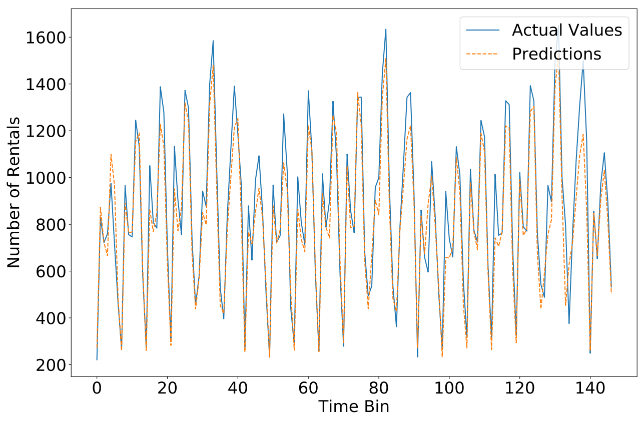


Figure 8.9: Short-term temporal prediction - Performance of the RFR model with expanding window in the validation set (24 days).

feature ranking. When training a tree, it is possible to compute how much each feature decreases the tree's weighted impurity. For a forest, the reduction in impurity from each feature can be averaged and the features can be ranked according to this measure. This gives a simple and interpretable feedback on which features are most useful for the prediction. We find that the most important features for the model are: (i) if we are in the daily peaks from 3 pm to 9 pm, (ii) during the night (0 am - 6 am) or (iii) if we are on a Friday and Saturday. Interestingly, the most important weather condition for the regressors is the presence of clouds, while the second one is a (rare) condition of presence of fog, mist and rain in the considered time bin.

8.5.4 The effect of weather information

At this point, it is relevant to discuss the importance of weather forecast for the predictions. First, for the long-term predictions, we did not use any weather information, as that would require perfect weather forecast in a period far in the future (in this case, three months). In order to validate the effect of weather in this idealized situation, we assumed such perfect forecast and evaluated the models using weather information as a feature. By assuming perfect forecast, we are able to set an upper bound on the effect of weather information on the models. The results show that, on average, weather information improved the models by about 3% on average.

Second, for the short-term predictions, we can use weather information. We assume perfect weather forecast in the short-term (next three hours). This assumption is reasonable once weather forecast for such short periods should be quite close to perfect. By doing so, we filter out any dependence on the particular weather forecast technique used (which could vary across different places/countries and is therefore out of the scope of this work).

According to the feature importance, among the features used for the short-term predictions (day of the week, hour of the day, and weather type), the weather is the least important feature. As such, we do not expect a great impact of weather mispredictions on the results. Indeed, the results with the random forests model (the one with the best performance among the models we evaluated) show that by removing weather information from the features the prediction accuracy decreases by less than 2% on the MAPE.

8.6 Spatial prediction of rentals with socio-demographic data

In this section I show the methodologies used to predict the demand of cars in a neighborhood without using past data as features. In other words, given only socio-demographic data in the neighborhoods, I try to predict the average number of expected rentals at each time bin, and at each neighborhood. This problem is often referred to as a green field or cold start approach. In this case, the operator is interested in knowing what could be the system usage in a new neighborhood (or even a new city) based only on socio-demographic data. Historical data are available from other neighborhoods (or cities), and are used only for training.

Since Vancouver presents a division in 22 neighborhoods which constitute the dataset for the training step, the analyses could suffer from an overfitting problem. To minimize this potential effect, I follow a state-of-the-art approach, namely leave-one-out testing:

given a target neighborhood, I consider information from all other neighborhoods for training the learning model, and consider the neighborhood that I left out for validation.

I manually select 83 socio-demographic features that might be related to human mobility. Here, I only apply the Support Vector Regression and Random Forest Regression models, given that they were the best performing models (aside from neural networks) in the temporal prediction. I do not consider neural networks since these are known to not work well with a very small training set as in this case. Additionally, being the RFR an ensemble method, it is known to be resilient to overfitting [78].

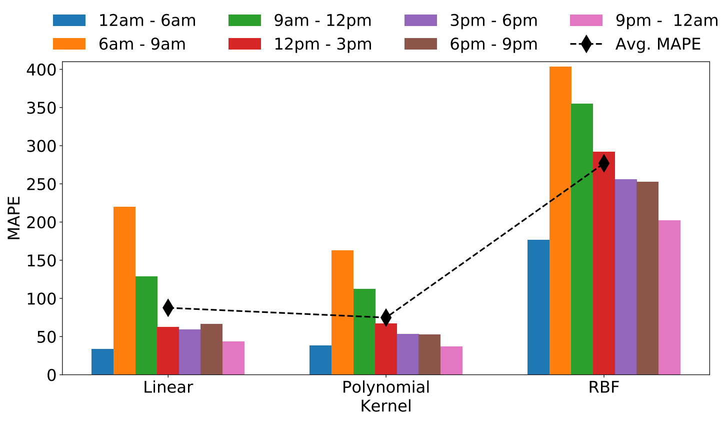
Considering hyperparameter tuning, for SVR, I try three different kernels (linear, polynomial and RBF), with different combinations of parameters. The best performances are obtained for $\epsilon = 0.1$, $C = 100$ ($C = 10$ for RBF), and $\gamma = \frac{1}{\#features}$ ($\gamma = 1$ for RBF). For RFR, I try number of trees ranging from 10 to 100. I show the impact of hyperparameter tuning in the following.

Figures 8.10a and 8.11b show the SVR prediction accuracy for the task of predicting the number of starting and ending rentals, respectively. For each kernel type and for each time bin, I report the average MAPE over the 22 experiments (one for each neighborhood that is left out during training). The SVR model performs rather poorly regardless of the parameter setting. Considering the targeted time bin, errors are higher for the morning slots, independently of the kernel, while the time bin from 0 am to 6 am is the one for which the model achieves the best performance. The polynomial kernel performs the best: yet the average (over all time bins) MAPE is 70% for the prediction of starting rentals, and 64% for the prediction of ending rentals. For the sake of completeness, best RMSE for starting and ending rentals predictions are 499.776 and 427.675, respectively, both for time bin from 0 am to 6 am.

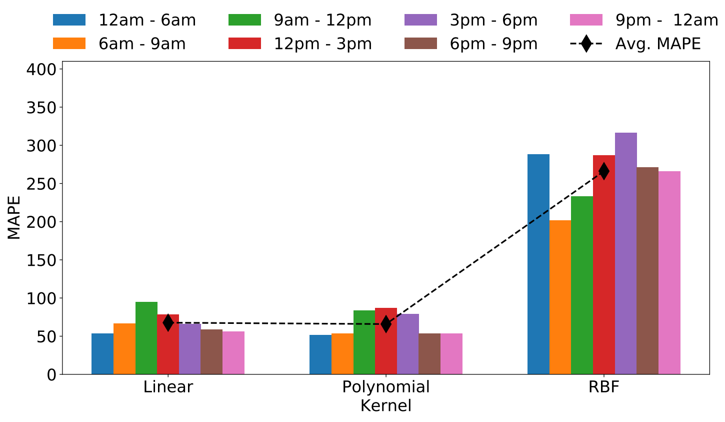
The results for the Random Forest Regression model are shown in figures 8.11a and 8.11b, for different number of trees. For a given time bin, I observe limited variation in the MAPE for increasing number of trees, which suggests that a small number of trees (30 or 40 trees, for instance) could be enough. This is expected given again the limited number of samples for the training. In this case, the overall MAPE is 59%.

Moving to the predictions for ending rentals in figure 8.11b, it is possible to observe smaller errors, with the best case with 20 or 40 trees, with the overall MAPE being 56%. Again, in the time bin from 0 am to 6 am I obtain the best predictions while the worst are obtained from 6 am to 9 am (for starting rentals prediction). Regarding RMSE measure, the best value for starting rentals is 427.260 for time from 0 am to 6 am and 50 trees, while for final rentals the best RMSE is 732.825 for time bin 6 am to 9 am.

Overall, the usage of only socio-demographic data as features offers from quite large prediction error. In the following, I show into which features are the most important so to also perform feature selection and possibly improve the model.



(a) SVR Model: Starting rentals



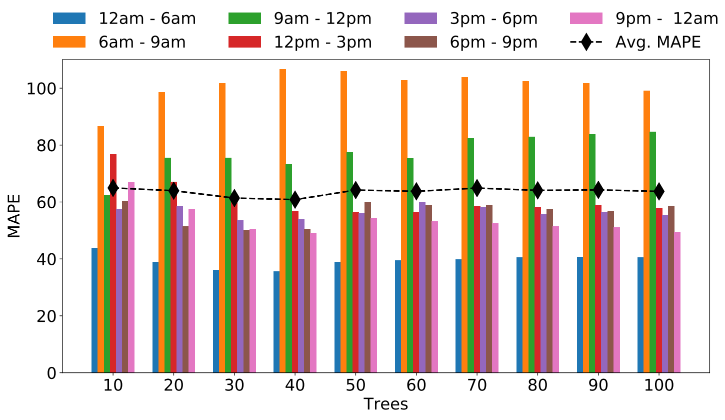
(b) SVR Model: Ending rentals

Figure 8.10: Spatial prediction - MAPE for Support Vector Regression models, using different kernels.

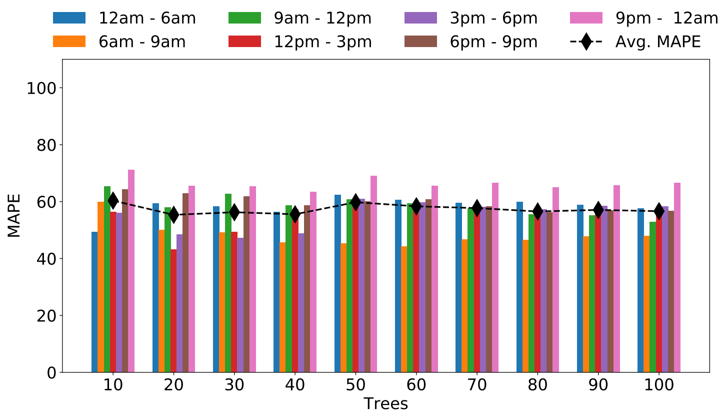
8.6.1 Feature ranking and selection

As in the previous section, I analyze the feature ranking for the RFR model. Table 8.3 reports the top-15 most relevant features. This feature ranking procedure allows us on the one hand to identify what information the FFCS operator should focus on when considering new neighborhoods of the city in which to implement its service. On the other hand, such ranking leads to a reduction of the number of features the model: in this way it is possible to focus only on the most important ones.

To evaluate the impact of the features on the model performance, I train once again the RFR with an increasing number of features, chosen according to the given rank. I fix the number of trees according to the best average MAPE obtained in Figures 8.11a and 8.11b: 40 trees for the starting and 20 for the ending rentals prediction. Figure 8.12



(a) RFR Model: Starting Rentals



(b) RFR Model: Ending Rentals

Figure 8.11: Spatial prediction - MAPE for Random Forests Regression models, using different number of trees.

shows the results. It reports the MAPE versus the number of features in the model. Notice the U-shaped curve of the average MAPE (dashed black line). Intuitively, too few features worsen the regression performance due to lack of information. Too many features also reduce the performance since the training is more complicated and the model gets confused.

I further evaluate the RFR model by selecting the best number of features (the one that minimizes the average MAPE), which results to selecting the top 7 features in Table 8.3. With this subset, the average MAPE is 41% and RMSE equals to 1104.501 for starting rentals, while for arrivals MAPE is 39% and RSME is equal to 1010.453. As expected, using only the top most important features improves significantly the performance.

Finally, I explore the spatial prediction error, i.e., I look if there are neighborhoods that present significantly higher errors than others. Figure 8.13 depicts the heatmap of the MAPE per neighborhood, averaged over all time bins. The more the area is red the higher the average MAPE is. Each green dot represents actual positions of starting or arrival rentals as recorded in the original trace. The areas having the highest error are the ones labelled 15, 18, 11 and 0. The neighborhoods 15, 18, and 0 are in the periphery and intersect only partially with the rental area of the FFCS operator. This mismatch confuses the prediction since the model assumes the operative area coincides with the total area of each neighborhood. Thus, the model predicts much higher numbers of rentals (reflecting the whole neighborhood area) than the ones that are actually done (reflecting the restricted operational area). Neighborhood 0 has instead a large presence of parks where clearly the car cannot operate. As such, the features of this area are also not reflecting the entire area, fooling the classifier.

In general, the performance of the spatial predictions is lower when compared to the temporal predictions. This is expected given the nature of the problem, the limited amount of available data, and because the number of rentals varies widely within each neighborhood. However, I would like to emphasize that the results of the spatial prediction are still quite useful: the ranking of the regions in terms of service demand is indeed preserved in the predictions. In other words, the neighborhood with the largest demands, which could be the preferred locations to extend the service, would still be predicted correctly.

8.7 Conclusions

In this paper, we studied the problem of predicting FFCS demand patterns in time and space, a relevant problem to an adequate provisioning of the service and maintenance of the fleet. Relying on data from real FFCS rides in Vancouver as well as the municipality socio-demographic information, we investigated to which extent modern machine learning based solutions allow us to predict the transportation demand.

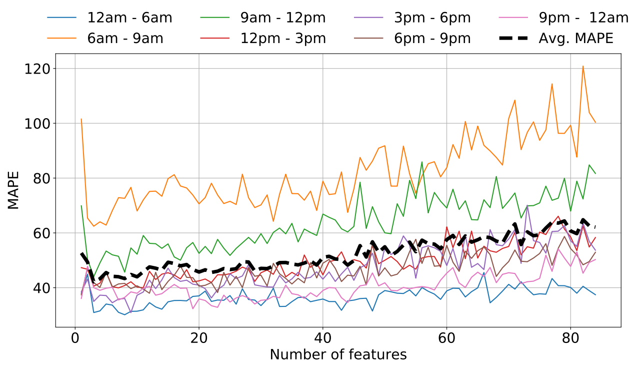
Our results show that the temporal prediction of rentals can be performed with relative errors down to 10%. In this scenario, a simple Random Forests Regression performs consistently among the best models, and allowing us to also discover which features are more useful for prediction. When considering the spatial prediction using socio-demographic data, we obtain relative errors around 40%, after feature selection. This is expected due to the scarcity of data, but the prediction results are still useful. Indeed, since the number of rentals varies widely within each neighborhood, the relative ranking is preserved. This is valuable for, e.g., look for the area where to first extend the service. Again, using a Random Forest Regression model, we can observe which features are the most useful for the prediction, a precious information for providers and regulators that wish to understand FFCS systems and to provide a high-quality service that benefits both providers and its costumers.

As future work, we would like to investigate whether this same strategy generalizes

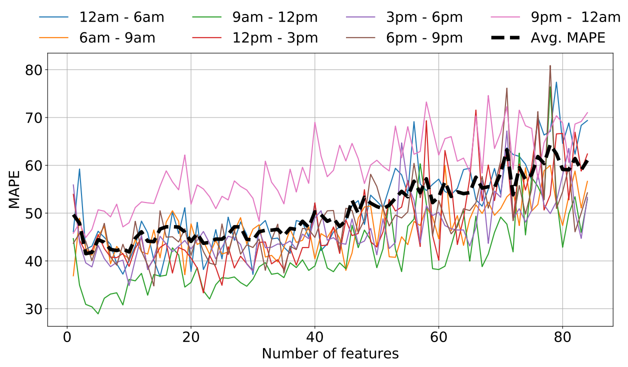
Rank	Feature	Relevance
1	Number of emergency calls	0.0717
2	Distance from downtown	0.0481
3	People commuting by walk	0.0381
4	People commuting within Vancouver	0.0342
5	People with income between 100 000 and 149 999 $CAD	0.0298
6	People with income between 60 000 and 69 999 $CAD	0.0286
7	People legally recognized as couple	0.0281
8	People with income more than 150 000 $CAD	0.0274
9	People divorced	0.0261
10	People commuting within the same neighborhood	0.0249
11	Couples having more than 3 children	0.0239
12	People with age between 50 and 54 years	0.0233
13	Unemployed people	0.0231
14	People never married	0.0217
15	People with income between 80 000 and 89 999 $CAD	0.0211

Table 8.3: Spatial prediction - Most relevant features and their importance for the prediction using Random Forest Regression. The first 7 are the ones that for obtain the best overall model

to different cities. Answering this question is challenging due to the heterogeneity and diversity of open data in different cities, and of usage patterns of car sharing around the world. We conjecture that given similar data the methodology could be applied to other cities, as there is nothing specific to the analyzed city in it. However the effectiveness of the models may change depending on peculiarities of each city. Still, it is an open problem towards which we have provided an important first step.



(a) Rentals starting



(b) Rentals arrivals

Figure 8.12: Spatial prediction - MAPE in the different time bins by selecting the most relevant features in RFR

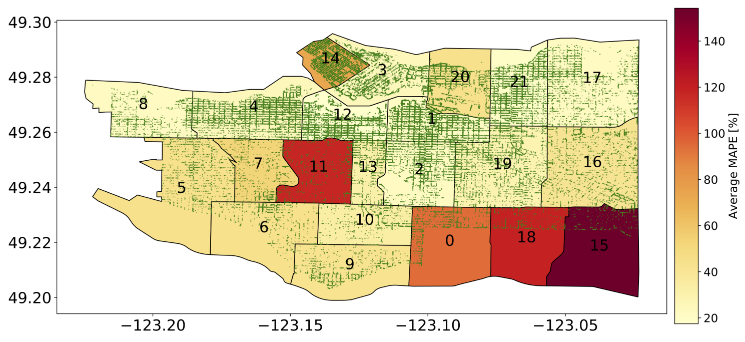


Figure 8.13: Spatial distribution - Heatmap of average MAPE per neighborhood. Rentals are shown on the map as green points

Chapter 9

Scalability of Electric FFCS in Smart Cities

This chapter is mostly taken from *"On Scalability of Electric Car Sharing in Smart Cities"* ([89]) published in the 2020 IEEE International Smart Cities Conference (ISC2) in September 2020.

9.1 Introduction

Today, around 55% of the world's population lives in urban areas, a proportion that is expected to increase to 68% by 2050 [1]. Cities face important challenges to manage mobility, with a mixture of public and private transportation means. The widespread usage of private cars led to land consume, increase of air pollution and higher health risk [90]. Private cars are often chosen by citizens for their flexibility and comfort, with the burden of higher fixed and variable costs. Recently, the sharing economy has brought regulators and policy makers to invest on free floating car sharing (FFCS) systems, car rental models where the customers can freely pick and drop a car within an operative area through a mobile app. They pay only for the time spent driving, usually with minute-based fares which include all costs. Thus, this combines some of the benefits of public transport and private cars [13]. Sharing the same car among different people helps reducing the number of vehicles and brings benefits for the whole community like increase of parking availability and reduction of pollution [91]. Moreover, since there are no fixed costs for users, usually FFCS is economically convenient for users who travel few thousand kilometers per year [92].

To make another step towards sustainable mobility, the challenge is to convert FFCS fleet from internal combustion engine vehicles (ICE) to electric ones (EVs), maintaining the same service flexibility. This change would further reduce the noise and pollutant emissions in congested areas [93], but calls for the creation of a charging infrastructure, and the management of the additional costs to handle battery charging operations.

In this work I analyze the feasibility and scalability of FFCS with electric vehicles. The goal is to find an economically sustainable solution that brings benefits to both citizens (i.e., high availability) and operators (i.e., high profit), with the economic sustainability being a crucial aspect. For instance in Italy, the main FFCS operator had revenues around 48 million euros in 2016, but still losing around 27 millions euros, with each car burning 4700 € on average[94]. Despite that, car sharing is estimated to increase from 20% to 40% from 2019 to 2021 [95].

However, the shift to EVs implies not trivial decisions due to the additional need of deploying and managing the charging infrastructure. What are its impacts on system performance and profit?

As a case study, I focus on the city of Turin in Italy. I leverage hundred of thousands of real FFCS trips [96] to extract the geo-temporal mobility demand. I use Kernel Density Estimation (KDE) to catch the demand spatial variability, and modulated Poisson models for the temporal demand [97]. I use it to feed a trace-driven flexible simulator that allows us to study how the design choices and system parameters impact on performance. I first consider an electric-car sharing system that has the same number of cars and faces same demand of the current one in Turin. I observe the impact of different charging infrastructure design, i.e., the number of poles and how to spread these are over the city area.

Next, I consider when the intensity of the mobility demand grows. How would the charging infrastructure need to grow correspondingly? And what is the impact of the fleet size? Summarizing the contributions, this paper proposes an answer to these questions making use of the demand model to project future or different scenarios and the cost-revenue model to evaluate the profitability of each configuration. I focus on performance indicators like the fraction of demand the system can satisfy and the total working hours it has to spend for the battery charging operations. Then, I project these into economical figures, observing how the design options impact on profitability.

The results show that the charging station placement is fundamental if poles are placed in areas with high demand, as cars get located where customers need them. This allows the system to naturally intercept the customers demand, thus to maximize the satisfied demand, and revenues. Considering system scalability, as expected the charging infrastructure must grow proportionally to the mobility demand. Interestingly instead, the number of vehicles can grow much slower, showing economy of scale savings which make the system likely profitable if well designed.

The paper is organized as follows: In section 9.2 I discuss this work in light of past literature for FFCSs and their economical aspects. After reporting the details about the dataset and demand model in section 9.3, and the simulator in section 9.4, I present results in section 9.5 before drawing conclusions in section 9.6.

9.2 Related Work

While first FFCS are operative since 2008 in Europe [29], the research on this topic has only recently flourished, especially for EVs. A common problem in transportation is to define models to optimize the fleet management and system design in general. For example, the authors of [98] proposed a Mixed Integer Problem (MIP) to maintain and organize fleet distribution in short term considering a stochastic demand. In the recent work [99] authors showed how to analytically model customers' probability to use car sharing. Other studies include more complex phenomena in their works, like [100] where the authors consider a non-linear charging function and detailed power lines constraints to optimally design *one-way* car sharing system (using a MIP).

Another strategy to study FFCS is to *simulate* how users interact with it. For example, the authors of [101] proposed (but did not implement) an agent simulator approach to measure how FFCS can be scaled on the entire Swiss traffic. The authors of [91] present a study of two-ways car sharing growth, with the help of an event-based simulator that measures if and how charging stations produce profits. Similarly, [102] proposes an open-source multi-agent simulator able to replicate travelling people's habits. In particular, it focuses on the realistic replication of users' behavioral model relying on multinomial distribution of modal choice.

Recently, the availability and abundance of data helped shaping FFCS users' behaviour. Considering this, some works like [103] and [96] scraped data from real ICE FFCS, characterized their services and proposed model generalizations. Big Data approaches helped researchers to improve the simulation fidelity. In particular, authors of [104] used data to predict the shareability of an urban ride, finding that this a property city-invariant. On the same optic, the authors of [105] use data from several American cities to optimize the position of the charging stations of a one-way car sharing, finding that the optimal results place the stations in high-demand areas. This results in this chapters confirms the results in [106] and [107] where I used a simulation based approach to measure the impact of different design options of an EVs FFCS system. Here, the use of big data to derive realistic demand models to feed accurate simulations. I move one step forward - showing that this is beneficial also as a proxy of relocation, i.e., cars get naturally relocated to high demand zones.

The economic sustainability is another key aspect of car sharing - especially with EVs. Authors of [108] studied the economic sustainability of one way electric car sharing systems finding out that charging station should have an amortization period of at least 5 years to produce profits. The authors of [109] compared how FFCS with EVs and ICE can produce profits, observing the best compromises with ICE fuelled with cheap and cleaner fuel like ethanol. Another study [110] concerning the city of Lisbon found out that switching to EVs would cost more than ICE and would lead to a negative profit of about one million per year. This is largely due to the higher cost of electric cars and of the charging infrastructure. This chapter explores which are the most efficient and economically sustainable combinations of fleet size, charging infrastructure design, also in light of demand growth.

To the best of I can tell, this work is among the first to study the scalability of a FFCS system with electric vehicles, exploring key parameters like number of poles, fleet size and increase in demand can affects economic and performance of the system.

9.3 Dataset and demand model

9.3.1 Dataset

In this chapter I leverage the data collected chapter 2 and described in chapters 3 and 4, capturing real trips performed by car2go users. These data let us model the users' mobility demand in time and in space. From this, I derive a demand model that generalize the users' demand observed in the real data. I use it to generate realistic traces describing possible user trips and feed them to the event-based simulator to derive performance figures.

Recalling that the dataset consists in actual rental performed by car2go in Turin composed by geo-temporal coordinates recording where and when a user's ride starts and end. Figure 9.1 summarizes the number of daily rentals from June 2017 to January 2018 in Turin, the reference dataset[1] and table 9.1 outlines the main characteristics of this data. 400 cars were available, travelling on average less than 4 km in each trip, for an average rental time of 21 minutes.

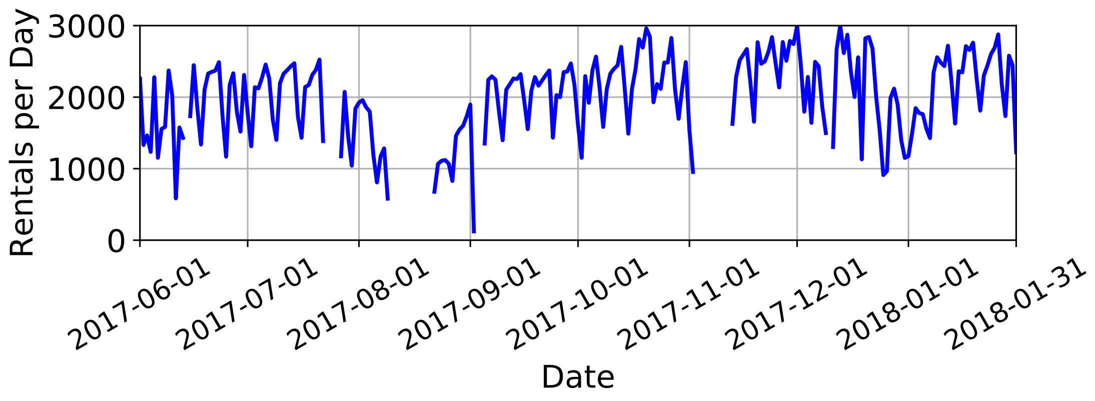


Figure 9.1: Number of rentals per day in Turin, from June 2017 to January 2018. Some data is missing.

Instead, directly use the original trace to observe system performance, I decided to create a model to observe what-if scenarios, e.g., to observe the impact of a growth in

[1]For some periods the crawler did not record data due to server failures.

Table 9.1: Main dataset characteristics, recorded in Turin from October to December 2017. Rental time and rental distance report both median (Med) and average (Avg) values.

Rentals	Fleet Size	Rental Time [min]		Rental Dist. [km]		Zones
		Avg	Med	Avg	Med	
180k	400	21	20	3.96	3.36	279

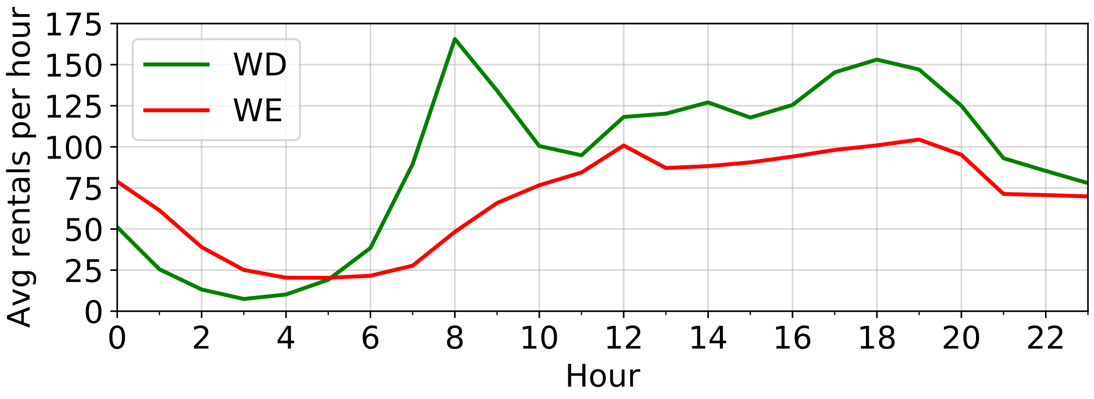


Figure 9.2: Average number of rentals per hour during weekdays (WD) and weekends (WE).

the demand. For this, I use the available data to create a generalized demand model. I follow the approach presented in in [97]. In a nutshell the demand is modelled in time by using modulated Poisson processes - a common accepted model for independent service requests of a very large population [111]. To capture the spatial heterogeneity, I generalize the traces using Kernel Density Estimation (KDE) [112]. KDE gives us the possibility to smooth the real data over a multi-dimensional space while maintaining the origin/destination correlation. In more details, for the request arrival time process the model assumes that the inter-arrival time of trips follows an exponential distribution with rate depending on the type (weekend or working day) and hour of the day. The model considers 24 time bins of 1 h each - 48 periods in total. In each time/day bin, the Poisson arrival rate matches the average rate of requests in that time bin in the original dataset as shown in figure 9.2. This temporal model allows to scale the overall demand by introducing a global scaling factor λ as a multiplier of the request rate of each time bin.

To model the spatial diversity of the demand, the models works on the case of study city divided in a set Z of contiguous 500 m x 500 m zones, obtaining in total 279 zones.

Each couple of spatial coordinates in the city area (x, y) maps to one and only one zone. Since each trip i departs from a certain zone (origin O_i, described by two coordinates) and arrives to another zone (destination D_j, described by two coordinates), it is therefore characterized by two couples of coordinates, that can be represented as 4 scalars. For each time bin, I derive an origin and destination matrix counting how many trips were originated from a given zone O and destined to a given zone D. Thus, in order to model the OD matrix in each temporal slot, I fit a 4-dimensional KDE based on the aforesaid coordinates. For each of these matrices, I compute a KDE model, using Gaussian kernels, with bandwidth equal to 1. Not reported here for the sake of brevity, I compare the number of trips generated from the model and the ones presented in the original trace. As expected, there is a very good match with low residuals. I refer the reader to [97] for details.

Notice that I employ a single global scaling factor λ directly the temporal model to keep the spatial distribution of trips unchanged while increasing the request rate.

9.4 Simulator and System Parameters

Armed with the generalized demand model, I design and implement an event driven simulator, improving the software depicted in chapter 5 to study the EVs FFCS system. Here I detail the simulation model, the simulator assumptions, the performance metrics, and the cost model used to project system performance into economic figures.

9.4.1 Simulator and assumptions

I consider a fleet F of electric cars. As the old version, the simulator works on the city divided Z of zones of 500 m x 500 m each, where cars can be parked, rented, charged and returned. Car characteristics are the same as MY2018 electric Smart ForTwo, namely $B = 17.6$ kWh battery capacity and 15.9 kWh/100 km energy efficiency like the previous works. Each car is characterized by its location, status (i.e., available, rented, under charge) and battery State of Charge (SoC). At simulation startup, cars are randomly placed in zones and as first new feature, the initial SoC uniformly distributed in $[0.5B, B]$, and marked as available.

The charging infrastructure considers n_p Level-2 chargers, with 3.7 kW nominal power and 92% charging efficiency. Charging stations are spread around the city zones. The simulator places poles in those zones having the highest probability of being destination zones. This results in a good strategy to maximize system performance [106, 107]. In details, the algorithm sorts zones $z \in Z$ by the total number of parkings $tot\_park(z)$ observed in the original trace. The simulator then consider the top z_p fraction of zones, and place a number of poles in each proportionally to

$$z_p \simeq n_p \cdot tot\_park(z) / \sum_z tot\_park(z) \tag{9.1}$$

At $t = 0$, the simulator generates the first *rental request* event, extracting origin and destination coordinates according to the KDE model of the current hour/day slot, and schedules the next rental request event according to the modulated Poisson process. Events are then processed as follows:

Car request event. When a *rental requests* fires, a customer looks for a car within the origin zone and in 1-hop neighbouring zones. If at least one car with enough SoC to reach the desired destination exists, the car gets rented, and a *car release* event is scheduled after the time to reach the destination that is proportional to the distance, considering both orography and road network shape [107]. If more than one such cars exists the closest one is picked, and, if need, the one with highest SoC. If no car is suitable for this ride, the trip does not occur and the request is marked as *unsatisfied*.

Car release. When a *car release* event fires, the simulator updates the car SoC decreasing it proportionally to the travelled distance. If the updated SoC is above a threshold α, the car is parked in the user's arrival zone, and marked available for other rentals.

If instead the SoC is below α, the car battery needs to be charged. The system handles the charging event by moving the car to the nearest-free charging pole. The simulator schedules a *charge complete* event which accounts for both the time to reach the pole and the time to bring the SoC to 100%[2].

Charge complete. When a *charge complete* event fires, the car is marked as available, and customers can rent it again. The charging pole is released as well. Notice that I assume the car is released in the same zone where it was being charged, i.e., the system does not implement any relocation policy after charging.

9.4.2 Performance metrics

In this chapter, I describe on the following performance metrics to compare different design options:

Unsatisfied Demand: it is the fraction of requests that are not satisfied because there is no car with enough SoC in the origin and neighbouring zones. It is an indicator of the quality of the service in terms of car availability for user requests, and shall be minimised.

Total charging handling time: it measures the monthly time spent by the system to bring cars to the charging stations. It is the sum of the driving time spent by workers to drive the cars to the nearest-free pole. It gives an indication of the goodness of the charging infrastructure. Being it a cost, it shall be minimized (see the operating costs described below).

[2]For simplicity, the assumption is the presence of infinite workers to handle the battery charge events so that a car gets serviced immediately. In case all poles are busy, the car gets placed in a queue of the closest charging station, and gets serviced when the first pole is freed.

Table 9.2: Summary of parameters and economic cost assumed for Turin.

Parameters used for the simulations		
Param	Description	Range
$\|F\|$	Fleet size	[80, 2000]
$\|Z\|$	Number of 500 m x 500 m zones	279
B	Battery capacity - Electric Smart ForTwo	17.6 kWh
n_p	Number of charging poles - 3,7kW each	[8, 300]
z_p	Fraction of zones with charging poles	[0.003, 0.20]
α	SoC charging threshold	0.25
λ	Rental demand rate scaling factor	[1, 5]

Cost and revenue parameters with values for Turin		
C_{lease}	Yearly electric Smart ForTwo vehicle lease cost	4000 €/yr/vehicle [113]
C_{pole}	Material cost of a level-2 charging pole	1700 €/pole [114]
C_{labor}	Labor cost to install a charging pole	2200 €/pole [114]
C_{setup}	Make-ready infrastructure cost per charging station	1500 €/station [114]
p_{life}	Charging station and pole lifetime - amortization period for C_{pole}, C_{labor} and C_{setup}	10 yr [110]
C_{maint}	Yearly pole maintenance cost	500 €/yr/pole [114]
C_{ground}	Yearly ground occupation tax	355 €/yr/pole [115]
C_{energy}	Energy cost for kWh	0.19 €/kWh [116]
$C_{drivers}$	Hourly labour cost to bring the cars to charge	23 €/h [117]
C_{disinf}	Disinfection and interior cleaning cost	15 €/charge [118]
C_{wash}	Cost to wash the car	8 €/100 rentals [118]
R_{rental}	Average revenue per rental minute (exl. VAT)	0.20 €/min [119]

9.4.3 Cost model

While performance indexes are useful to explore design options, the FFCS operator is ultimately interested in the economic sustainability of a solution. For this, I derive a cost model based on yearly projections. I then consider revenues by projecting the number of rental and their duration. Armed with both, I estimate profit. Here I consider:

Vehicle cost. I assume cars are leased to include all costs, namely registration, tax, insurance, ordinary and extraordinary maintenance, and roadside assistance. I assume electric cars do not pay for parking on street and for accessing limited traffic areas. Given the yearly car lease C_{lease} and the number of vehicles, I easily derive the total yearly fleet cost.

Charging infrastructure cost. Here I refer to actual use cases as defined in [114]. Pole installation costs account for material and labor cost. Material cost C_{pole} includes hardware cost for Level II chargers. Labor cost C_{labor} is highly dependent on the city, region and country. I need also to consider the make-ready infrastructure cost C_{setup} that represents the cost for a charging station setup. It does not depend on the number of charging poles per station, but depends only on the number of charging zones $z_p \cdot |Z|$. It represents a highly variable cost since it depends on the location and the electric distribution infrastructure already in place. In fact, the expenses of trenching and laying conduit can add thousands of Euros to costs. All these costs are one-time costs. I assume these costs have an amortization period equal to the average charging station and pole lifetime p_{life}.

Next, I consider pole maintenance costs C_{maint}, which I derive from variable site-specific parameters.

In some cities, I need also to consider the per vehicle ground occupation tax C_{ground}, that usually depends on the surface for dedicated charging spot. Due to the small size of Smart ForTwo, charging spots are assumed to be 4,50 m x 2,30 m, for each pole.

Operating costs. For this I take into account the C_{energy} cost for the energy to charge the vehicles; the hourly cost for workers $C_{drivers}$ who have to handle the charge events; a cost C_{disinf} to clean and disinfect the car any time the worker brings it to charge. Finally, I assume exterior car washing every 100 rentals, each costing C_{wash}.

Rental Revenue I consider a simple average cost-per-minute R_{rental}. This allows us to transform the total rental minutes into the total revenues.

Top part of table 9.2 summarizes the parameters that define the scenarios used in the simulations. Bottom part shows the cost I consider for the Turin use case. Given a scenario, I run a simulation to collect performance indexes. I next post-process the simulation results to derive the monthly cost and revenue figures. The custom simulator used is written in Python and based on SimPy library.[3] The cost-revenue model is implemented in Python too and it is available online. The cost model allows one to interactively observe what happens by changing the cost values.[4]

9.5 Results

Given the multiple system design parameters, here I proceed by steps. First I analyse the impact on performance in order to select good design options. I then project the results through the cost figures to gauge the economic implications of these choices.

I consider as starting parameters the ones referring to the current FFCS running in Turin based on ICE cars, i.e., a fleet size $|F| = 400$ and demand scaling factor $\lambda = 1$. I explore the charging infrastructure design options, namely its size n_p and extensiveness z_p. I fix $\alpha = 0.25$ corresponding to the minimum energy needed to perform the longest trip in Turin [106]. I also check the impact of increasing the demand up to $\lambda = 5$. Correspondingly, I increase the fleet size $|F(\lambda)| = 400 \cdot \lambda$ by the same factor. Each simulation considers three months of virtual time, corresponding to more than 200 000 rental requests for $\lambda = 1$.

[3]SimPy is a discrete-event simulation library. Documentation is available at: https://simpy.readthedocs.io/en/latest/contents.html.

[4]The code and data for cost and profit evaluation are available at: https://smartdata.polito.it/on-scalability-of-electric-car-sharing-in-smart-cities/.

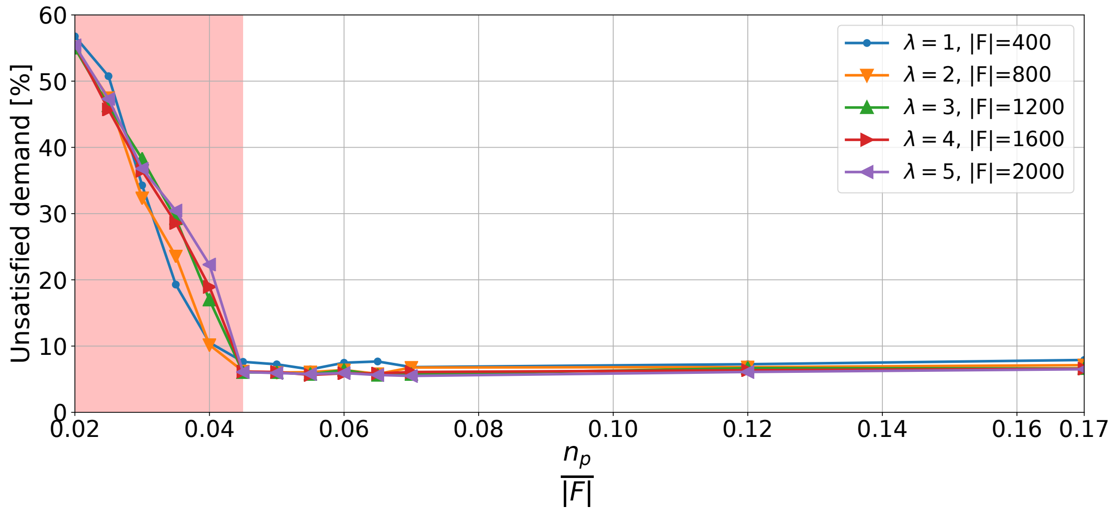


Figure 9.3: Unsatisfied demand with respect to different number of poles per vehicle. Curves show performance with different demand factor λ and fleet size $|F|$, with z_p = 0.05.

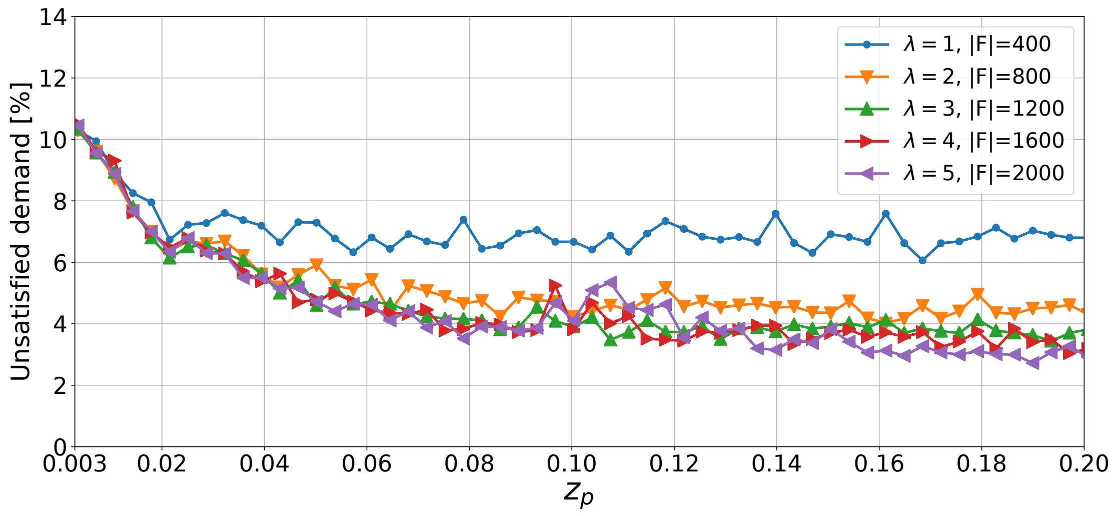


Figure 9.4: Unsatisfied demand with respect to the fraction of zones with charging poles z_p. Curves show performance with different demand factor λ and fleet size $|F|$, with $n_p/|F| = 0.06$.

9.5.1 Impact of infrastructure design options

Focus first on the impact of the number of poles per vehicles $n_p/|F|$ on the unsatisfied demand - reported in figure 9.3. Consider $\lambda = 1$ first. Here $z_p = 0.05$ (14 charging zones). I observe two working regions: on the right - the charging infrastructure has enough capacity to supply the energy to support all customers' trips - resulting in a constant unsatisfied demand. On the left, system charging capacity goes below a minimum threshold (highlighted by the red area). Here the charging infrastructure cannot supply enough energy and unfeasible trips grow linearly with the lack of energy. Consider now a demand factor that doubles ($\lambda = 2$, and $|F| = 2 \cdot 400$). The energy supply must grow by a factor of 2 to supply twice the number of trips. As such the minimum threshold in terms of number of poles per vehicles remains the same. The same holds for higher λ. Interestingly, the number of poles to supply the energy to cope with the mobility demand is quite small: a pole every 20 cars results enough.

There is still a 5-7% of unsatisfied demand which results from the mismatch between zones with available cars, and zones with demand. I now check the impact of z_p on this. figure 9.4 fixes $n_p/|F| = 0.06$, and shows the impact of concentrating or spreading them on few or more zones. On the leftmost case, I have the "charging hub" scenario, meaning that all poles are located in a single zone where all cars must be brought for charging. This solution creates a surplus of cars in the zone where the hub is, and a lack of cars in other zones. Unsatisfied demand then grows, calling for relocation policies. Increasing z_p has the benefit of spreading cars in the city[5]. Having opted to place poles in top_park zones, cars get naturally located there, facilitating customers that look for a car in those high-demand zones. This reduces the percentage of unsatisfied demand significantly, questioning the need of costly relocation policies. Performance-wise, the higher the fraction of zones with poles, the better.

Focus now on the time the system has to spend to bring cars to the closest charging pole, reported in figure 9.5 for $z_p = 0.20$. Notice that if the system cannot supply enough energy to satisfy the demand ($n_p/|F| < 0.055$ in this case), the charging handling cost decreases. Likely not a good design choice being this due to loss of satisfied demand. Consider the region $n_p/|F| \geq 0.055$, where the system has enough charging capacity. If there are just enough poles, most of them results busy, and the workers need to drive cars to far away free poles. This results in an increase of handling time up to 1600 h for $\lambda = 5$. By increasing $n_p/|F|$, I increase the probability of finding a nearby free pole, shortening handling time down to 1000 h per month for $\lambda = 5$. As expected, the higher λ, the higher the time to handle charging events – with an almost perfect linear increase.

This highlights a trade-off between infrastructure costs and management costs. To better gauge this, figure 9.6 compares a more concentrated system with $z_p = 0.05$ with a more distributed system with $z_p = 0.20$. I plot the "additional satisfied demand" (black curve) and the saving in charging handling time (red curve) for increasing demand factor λ and fleet size $|F|$ (fixing $n_p/|F| = 0.06$). In all cases, $z_p = 0.20$ results in higher satisfied demand and lower cost than $z_p = 0.05$, with benefits that increase with increasing demand. In a nutshell, distributing the same number of poles among more zones improves system performance and reduces charging handling time. Clearly this

[5]For $\lambda = 1$ I can equip $n_p = 24$ zone maximum ($z_p = 0.09$), after which results do not change. Differences are due to simulation randomness.

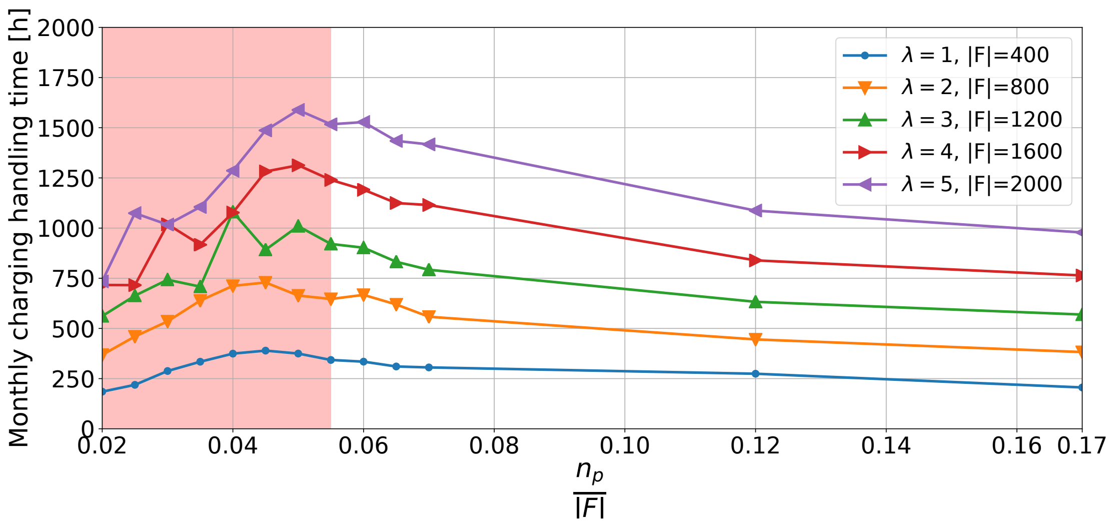


Figure 9.5: Monthly charging handling time drivers have to spend to bring cars to charging poles. Curves show performance with different values of demand factor λ and fleet size $|F|$, with $z_p = 0.20$.

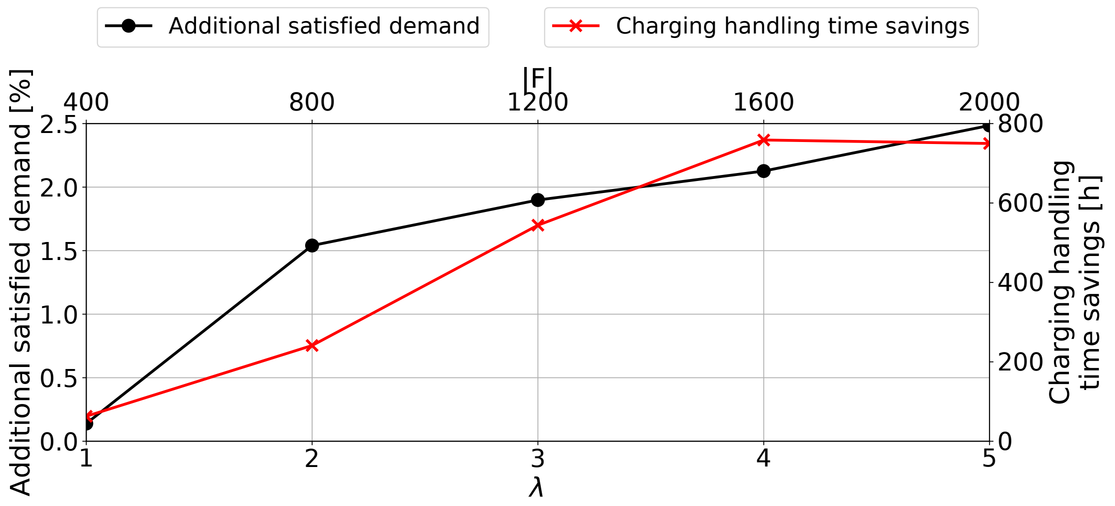


Figure 9.6: Gain in terms of satisfied demand and charging handling time of a more spread infrastructure ($z_p = 0.20$) with respect to a more centralized infrastructure ($z_p = 0.05$). Gain is showed for increasing demand factor λ and fleet size $|F|$. Here $n_p/|F| = 0.06$.

needs to be weighted by the additional cost of installing a more distributed charging infrastructure.

9.5.2 Impact of fleet size

I now explore the impact of the number of cars when the demand increases. For this, I scale λ, but keep the fleet size constant to $|F| = 400$. For simplicity, I consider a charging infrastructure capacity that can cope with the highest demand, i.e., $n_p = 120$. I consider two cases, $z_p = 0.05$ and $z_p = 0.20$. I plot results in figure 9.7. Interestingly, the same number of vehicles can sustain a sizeable increase in λ without a significant impact the unsatisfied demand. For instance, $|F| = 400$ vehicles can cope with a factor $\lambda = 3$ increase in the demand, just losing 2.5% of customer requests. This would result in a significant saving in the cost of vehicles. Given there are no differences in concentrating or spreading the charging poles in few or more zones, I fix $z_p = 0.20$ from now on.

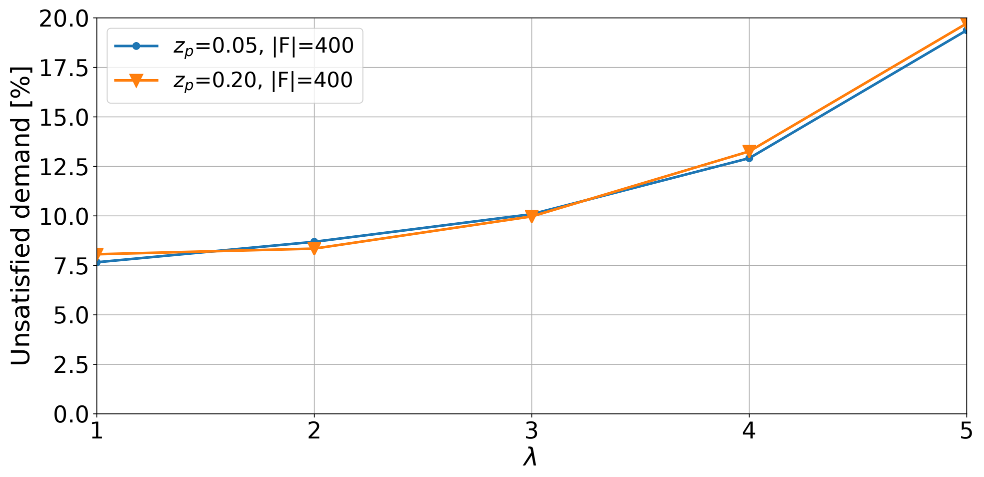
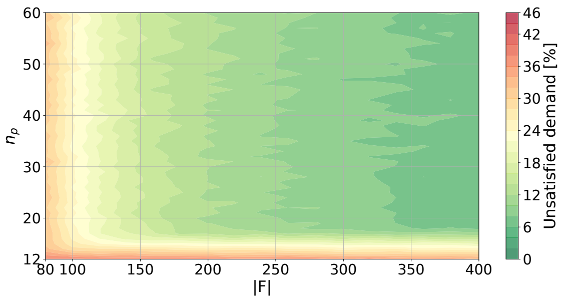


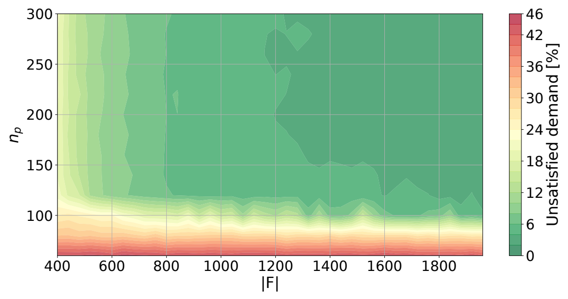
Figure 9.7: Unsatisfied demand for increasing values of λ but same number of vehicles $|F| = 400$, and large enough charging infrastructure ($n_p = 120$).

To observe how the unsatisfied demand would be impaired by further reducing the number of cars and/or the number of poles, I present contour maps in Figures 9.8a and figure 9.8b, for $\lambda = 1$ and $\lambda = 5$, respectively. Interestingly, the two design parameters seem to affect unsatisfied demand in almost independent manner. On the one hand, reducing the number of cars has limited impact until I reach very small values. For instance, halving the fleet size down to $|F| = 200$ would increase the unsatisfied demand by 6-8% only. On the other hand, increasing n_p brings no benefit - provided there are enough poles (cfr. figure 9.3).

The same considerations hold when $\lambda = 5$, with a slightly higher interaction between n_p and $|F|$ when approaching small values for both. Observe also a large region with unsatisfied demand lower than 6% (dark green). The high number of vehicles allows a high multiplexing gain so that fewer cars can offer the same service level. For instance, $|F| = 800$ cars guarantee about 10% of unsatisfied demand if $n_p > 120$. In a nutshell,

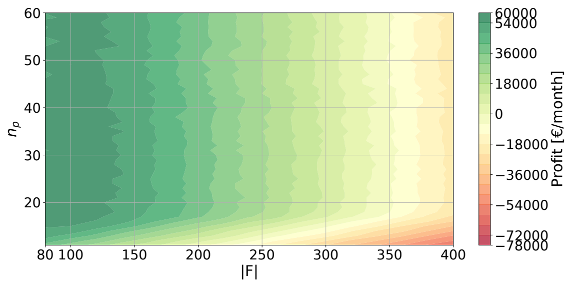


(a) Demand factor $\lambda = 1$

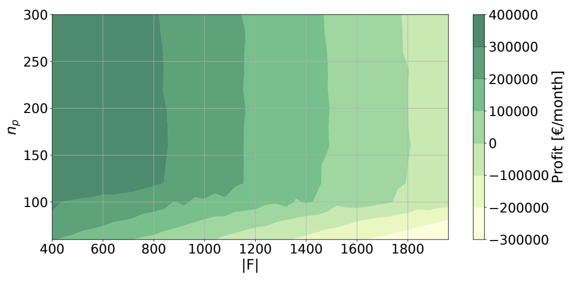


(b) Demand factor $\lambda = 5$

Figure 9.8: Unsatisfied demand varying number of poles n_p and fleet size $|F|$ with $z_p = 0.20$.



(a) Demand factor $\lambda = 1$



(b) Demand factor $\lambda = 5$

Figure 9.9: Monthly estimated profits varying number of poles n_p and fleet size $|F|$ with $z_p = 0.20$.

the system needs less cars to satisfy the same percentage of demand when the demand increases, with significant economy of scale gain.

9.5.3 Impact of costs

To have a clear and complete picture, I now project the performance indexes into economic figures. Here I compare the monthly profit an EVs FFCS provider would reach for different combinations of the number of vehicles and the number of poles, i.e., its investment in the fleet and charging infrastructure. All costs included in Table 9.2 are considered.

figure 9.9a shows the results for $\lambda = 1$. Green shades reflect positive profit, while yellow and red shades highlight loss-making configurations. Interestingly, the zones with the highest profits tend to be in the leftmost part of the figure, i.e., for small number of cars. While this causes a higher unsatisfied demand - see figure 9.8a - it looks the only way to reduce the cost of the fleet so to have a profitable system. The impact of the charging infrastructure is quite negligible unless when n_p becomes too small (i.e., when not enough charging capacity is present). This is due to the low cost of buying and installing a charging pole when amortized on $p_{life} = 10$ years. Recall that I have seen that the charging infrastructure design calls for a charging pole every 20 vehicles. As such, the overall economic impact of the charging infrastructure results quite negligible compared to the fleet size costs.

The picture improves drastically when $\lambda = 5$, shown in figure 9.9b. Here, I explore scenarios with a 5-fold increase in both the number of poles ($n_p \in [50{,}300]$), and in the number of vehicles ($|F| \in [400{,}2000]$) with respect to the $\lambda = 1$ scenario. Here, all configurations result in positive profit. Even more interestingly, by reducing the number of cars I observe a marginal decrease in the profit, with the best scenarios being in $|F| \in [1400, 2000]$. This is due to large multiplexing effect I already observed in figure 9.7. Even when reducing the number of cars to 800, I observe sizeable profits.

Considering the number of poles, as expected, when $n_p < 120$ (the minimum threshold for constant unsatisfied demand when $|F| = 2000$, showed in figure 9.3) the insufficient system charging capacity impairs cars availability, increasing the unsatisfied demand. As seen already, increasing n_p above the minimum threshold brings little benefit, but it also has little impact on the profits (due to the relatively low cost of pole installation).

In summary, I can conclude that the FFCS provider needs to carefully evaluate the minimum number of poles when designing and implementing the charging infrastructure. The limited cost of pole installation, and the long amortization time make over-provisioning the charging infrastructure a viable option to make the system robust to demand increase. Considering the fleet size, when the demand is low, the high cost of vehicles suggests limiting the number of vehicles. When instead the demand grows, an economy of scale gain is possible, making the system even profitable with large fleet size.

9.6 Conclusions

In this chapter I presented a simulation study of free floating car sharing systems. Armed with a realistic demand, I studied the performance implications of moving from ICE FFCS to a EV based solution.

This study offers several take-away messages: first, the charging infrastructure must be able to provide enough energy to cope with the mobility demand. Interestingly, it results to be quite limited, with just 20 poles able to sustain a system of 400 vehicles. Second, distributing the charging poles in zones where cars get frequently parked and rented is instrumental to maximize the demand the system can satisfy, while also limiting the time workers have to spend to bring cars for charging. Third, the system exhibits useful economy of scale, so that the fleet size shall increase sublinearly with respect to the mobility demand intensity.

At last, when projected into economic figures, the fleet setup and management represent the main cost factors. Choosing the right number of vehicles results more fundamental than optimizing the charging infrastructure costs. For instance, for the current demand intensity in Turin, the switch to EVs must be carefully designed to be profitable. Interestingly, when the demand grows, the margins are much higher, allowing some nice economy of scale opportunities.

As future direction, I are studying different cities as new use cases, looking at opportunities of involving users in the charging process in order to decrease management costs.

Chapter 10

Conclusions and Future Works

In this thesis, I proposed a data-driven pipeline to study the conversion from internal combustion engine electric vehicles in FFCS. Strongly relying on real FFCS data, optimization algorithms, and machine learning techniques, I depicted a pipeline able to evaluate the performances of electric FFCS in hot and cold start-up scenarios.

In the first part of my thesis, I described the data gathering, scraping real users' ride from operative combustion engine FFCS. Then I characterized the users' habits in 2017 for two FFCS providers: car2go and Enjoy. I highlighted the difference between one-way CS, two-ways CS, and FFCS, which pointed out how the users prefer flexibility over costs. Once all the data are consolidated, I developed an electric FFCS event-based trace-driven simulator. It takes into input the list of rentals events, the infrastructure setups (in terms of charging station placements, car consumptions), and the provider's fleet management policies. It replicates users' patterns and returns as output the metrics describing system performance (complete battery depletion, car unavailability) and users' discomfort caused by car plugging procedures. Armed with this, I initially investigated the best electric FFCS setup varying case of study city (and thus users' patterns), finding how an electric FFCS is sustainable with few charging stations. Next, I improved the solution of the previous step by running several optimization algorithms maximizing both system resilience and minimizing the users' discomfort. Then, I moved my attention to users' ride predictability, computing how weather and socio-economics aspects are related to the users' trips. Finally, I simulated possible electric FFCS growth targeting which are the most sensible parameters, and how profitable an electric FFCS may be.

The results are surprising. In particular, the data-driven charging station placement showed how the current demand might be sustained with only 104 charging poles in a city of about 1 million inhabitants like Turin. This estimate may still be reduced with ad-hoc algorithms at 72 poles. Finally, the scalability studies (made open source) may lead FFCS providers and policymakers aware of all the benefits of electric mobility.

However, the research presents several limitations. In particular, the dataset is composed of passive measurements, the reason that it is impossible to distinguish users' trips and unallocated missed trips. The software does not really know when the users did not find an available vehicle nearby. Moreover, the real traveled distances are unknown due to the anonymization related to privacy. This should not really impact macro analyses, but it becomes sensible when it necessary to perform neighborhood-grained optimizations. This dissertation does not take into account fleet management (relocation). This topic opens several research threads that need a pragmatic formulation and sophisticated upgrade to the software. Finally, the recent COVID-19 pandemic changed drastically both private and shared users' patterns opening new questions on mobility sustainment in smart cities.

In general, I believe my thesis proposes a pragmatic methodology to study the electric (r)evolution of shared mobility. It will be very challenging and stimulating for me to face the unsolved problem in my future research career.

Appendix A

Simulator Structure

A.1 Introduction

In this chapter I provide high level details about the simulator structure. In particular I provide the module composition in A.2 where I describe the three fundamental simulator's entity: the Car, the Event and the Zone modules. After that, I provide implementation details about the the simulator workflow in section A.2.

A.2 Inputs

In this section I describe the implementation details of the most relevant inputs that the simulator depicted in chapter 5. There are three main that models the mobility in the simulator: the *car*, the *event* and the *city_tiles* collection.

A.2.1 Car module

The car module is the class that implements all the operation related to the car. The significant attributes of this class reported in table A.1 Intuitively, the car present several methods that set and get parameter to e from the object instance. In particular, the by varying the consumption it is possible chose another car model.

A.2.2 Event module

This module represents the two possible events that trigger cars' status and moving trajectories. It contains as attributes make it possible distinguish between *rental start* and *rental end*, the geo-temporal coordinates. Keep in mind that the zone ID is obtained by joining the events coordinates into the 500m x 500m tiling procedure whom I divided the city. The Event temporal sequences represents the simulator's input trace. It exactly

Attribute	Description
ID	car identifier
startTime	timestamp when the rental starts
startChargeTime	timestamp marking the starts of plugging procedure
startZone	zone ID where the users rents the car
startEventID	event ID that makes the car busy
currentZone	zone ID of car current position
isBusy	flag stating when a car is rented
onCharge	flag stating when a car is plugged
soc	battery state of charge [%];
consumption	consumption [%kWh/km];

Table A.1: Car module attributes

replicate the users' demand in terms of temporal and spatial habits. Table A.2 reports the main attributes of the Event module.

Attribute	Description
ID	event identifier
type	*rental start* or *rental end*
timeStamp	timestamp marking the event shot
zoneID	zone ID where the event happens
nextPrevEventID	stores the related *Rental end* ID if the event is a *Rental start* and viceversa

Table A.2: Event module attributes

A.2.3 Zone module

This module models the 500m x 500m whom the city is divided during the simulation. It is useful to marks the tracks the cars available in a given zones and that can be used to satisfy the users' demand. The interaction between Car and Zone module replicates the users-car-FFCS system interaction. If a zone is desgined to host a charging station, other attributes are set in order to properly simulates the plugging procedures. Table A.3 reports the main attributes of the Event module, while table A.4 shows the attributes that are used when a zone hosts a charging station.

Attribute	Description
ID	zone identifier
availableCars	list of Car module IDs of all the cars parked in the zone
timeStamp	timestamp marking the event shot
neighbours	list of zone IDs corresponding to the cells in the North, South, West and East
isChargingStation	flags stating if the current zone hosts a charging station

Table A.3: Zone module attributes

Attribute	Description
numberOfPoles	total number of available plugs
busyPoles	list of car IDs representing the curren plugs
suppliedPower	[kWh]

Table A.4: Zone module attributes when the zone hosts a charging station

A.3 Simulator Core

In this section I illustrate how the main simulator logic blocks. The algorithm A.1 illustrates the main steps. All the input variables are global and accessible from the inner code. It is basically composed by a main loop that consumes the event set trace(line 2).

A.3.1 Main block

The block between lines 3 and 7 manages the users' car lookup procedure. In line 4 the the software looks for the closest car to the zone where the rental starts. In line 5 the simulator sets the parameters that records car status before the rental starts like starting position and new state of charge (if the car were plugged). Finally, line 6 moves the current car in the list of rented cars.

On the other hand, the block from line 8 to line 13 manages the users' rental end. In particular, line 9 performs the lookup procedure to retrieve the car involved in the rentals. Line 10 manages the car parking according the set policy. Line 11 updates the car state of charge, decreasing it proportionally to the travelled distance. Line 12, at the end of the ride computes the log that will be used to compute all the KPIs .

A.3.2 Functions

In this section I provide the logic behind the th two main function of algorithm A.1.

In particular algorithm A.2 illustrate the dynamics implemented while the a user look for a car when a *rental start* event is triggered. The block between line 2 and line 7 corresponds to looks for a car in the same zones of the event. If there are, the function

Algorithm 2: Main simulator structure

```
Input  : Events -List composed by rental start and rental end
Input  : Cars - List composed by cars elements representing the all the available cars
Input  : RentedCars - List composed by cars elements representing the all the rented cars
Input  : City - List composed by zones previously initialized with the charging station placement
Input  : policy - Policy controlling the car releases and plugging
Input  : w - users willing to collaborate with the system
Input  : π - security threshold for state of charge
1  for event in Events do
2      if (event.type == "start") then
3          car_id = getClosestCar(event.zoneID)
4          Cars[car_id].reserveCar(event)
5          RentedCars.append(Cars[car_id])
6      end
7      else
8          car_id = lookupCar(event)
9          Cars[car_id].updateSoC(event)
10         releaseCar(car_id, event)
11         computeMetrics()
12     end
13 end
```

Figure A.1: Pseudocode of the simulator

Algorithm 3: Structure getCloserCar()

```
Input  : event - rental start event
Output: car_id - ID of the closest car
1  listOfAvailableCars = []
2  for cars_id in City[event.zoneID].availableCars() do
3      listOfAvailableCars.append(cars_id)
4  end
5  if (listOfAvailableCars.length > 0) then
6      return getRandom(listOfAvailableCars)
7  end
8  else
9      for zone_id in City[event.zoneID].neighbours() do
10         for zone_id in City[zone_id].availableCars() do
11             listOfAvailableCars.append(cars_id)
12         end
13     end
14     return getRandom(listOfAvailableCars)
15 end
```

Figure A.2: Pseudocode of the simulator

returns one random cars. The lines from 8 to 14 implement, with the same logic, the car search in the neighbouring areas if the previous search did not produce any available car.

The algorithm A.3 shows the car return management. More in details, the lines of cod from 1 to 6 manages the the so-called Free-Floating policy: the users plug the car only if there are charging station and available spots in the desired arrival zone.

Conversely, line from 7 to 12 force the user to plug the car only when the car state

Algorithm 4: Structure of the releaseCar()

```
    Input : car_id - car to park
    Input : Event - rental end event
 1  if policy == "free-floating" then
 2      if City[event.zoneID].isChargingStation() &
 3      City[event.zoneID].busyPoles < City[event.zoneID].numberOfPoles then
 4          Cars[car_id].onCharge = True
 5      end
 6  end
 7  if (policy == "needed") then
 8      if Cars[car_id].soc < $\pi$ then
 9          zone_id = getClosestChargingStation(event.ZoneID)
10          Cars[car_id].reroute(zone_id)
11          Cars[car_id].onCharge = True
12      end
13  end
14  if (policy == "hybrid") then
15      userWilling = getRandom(0,1)
16      if (
17      City[event.zoneID].isChargingStation() &
18      City[event.zoneID].busyPoles < City[event.zoneID].numberOfPoles &
19      userWilling> w
20      ) |
21      (Cars[car_id].soc < $\pi$ ) then
22          zone_id = getClosestChargingStation(event.zoneID)
23          Cars[car_id].reroute(zone_id)
24          Cars[car_id].onCharge = True
25      end
26  end
```

Figure A.3: Pseudocode of car releasing

of charge is below the security π threshold. It is possible to notice how line 9 manages the rerouting towards the closest charging station so far as the designed arrival zones does not have available charging poles.

Finally, lines 15 to 25 describes the hybrid policy which combines the constraints of the needed and hybrid ones. In particular the random willingness selected in line 15 and tested in line 19 simulates the the condition on the users' selfishness. The higher the w value is the less probable is that the users decide to plug the car at the end of a ride.

Appendix B

List Of Publications

B.1 Journal Publications

- **Cocca, M.**, Giordano, D., Mellia, M., & Vassio, L. (2019). Free floating electric car sharing design: Data driven optimisation. Pervasive and Mobile Computing, 55, 59-75.
- Alencar, V. A., Rooke, F., **Cocca, M.**, Vassio, L., Almeida, J., & Vieira, A. B. (2019). Characterizing client usage patterns and service demand for car-sharing systems. Information Systems, 101448.
- **Cocca, M.**, Giordano, D., Mellia, M., & Vassio, L. (2019). Free Floating Electric Car Sharing: A Data Driven Approach for System Design. IEEE Transactions on Intelligent Transportation Systems, 20(12), 4691-4703.
- **Cocca, M.**, Teixeira, D., Vassio, L., Mellia, M., Almeida, J. M., & Couto da Silva, A. P. (2020). On Car-Sharing Usage Prediction with Open Socio-Demographic Data. Electronics, 9(1), 72.

B.2 Conference Publications

- Ciociola, A., **Cocca, M.**, Giordano, D., Mellia, M., Morichetta, A., Putina, A., & Salutari, F. (2017, August). UMAP: Urban mobility analysis platform to harvest car sharing data. In 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI) (pp. 1-8). IEEE.
- **Cocca, M.**, Giordano, D., Mellia, M., & Vassio, L. (2018, November). Data driven optimization of charging station placement for EV free floating car sharing. In

2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2490-2495). IEEE.

- **Cocca, M.**, Giordano, D., Mellia, M., & Vassio, L. (2018, June). Free floating electric car sharing in smart cities: Data driven system dimensioning. In 2018 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 171-178). IEEE.
- Ciociola, A.,**Cocca, M.**, Giordano, D., Vassio, L., & Mellia, M. (2020, September). E-Scooter Sharing: Leveraging Open Data for System Design. In 2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT) (pp. 1-8). IEEE.
- Barulli, M., Ciociola, A., **Cocca, M.**, Vassio, L., Giordano, D., & Mellia, M. On Scalability of Electric Car Sharing in Smart Cities. In 2020 IEEE International Smart Cities Conference (ISC2) (pp. 1-8). IEEE.

Bibliography

[1] United Nations. *Revision of World Urbanization Prospectsa.* Accessed: 2020-07-09. 2018.

[2] Anna Matas and Josep-LLuis Raymond. "Changes in the structure of car ownership in Spain." In: *Transportation Research Part A: Policy and Practice* 42.1 (2008), pp. 187–202.

[3] Luciana Leirião et al. "Environmental and public health effects of vehicle emissions in a large metropolis: Case study of a truck driver strike in Sao Paulo, Brazil." In: *Atmospheric Pollution Research* (2020).

[4] Jonathan I Levy, Jonathan J Buonocore, and Katherine Von Stackelberg. "Evaluation of the public health impacts of traffic congestion: a health risk assessment." In: *Environmental health* 9.1 (2010), p. 65.

[5] Mohammed Raza Mehdi et al. "Spatio-temporal patterns of road traffic noise pollution in Karachi, Pakistan." In: *Environment international* 37.1 (2011), pp. 97–104.

[6] Marianna Jacyna et al. "Noise and environmental pollution from transport: decisive problems in developing ecologically efficient transport systems." In: *Journal of Vibroengineering* 19.7 (2017), pp. 5639–5655.

[7] Matthias Sweet. "Does traffic congestion slow the economy?" In: *Journal of Planning Literature* 26.4 (2011), pp. 391–404.

[8] Kent M Hymel, Kenneth A Small, and Kurt Van Dender. "Induced demand and rebound effects in road transport." In: *Transportation Research Part B: Methodological* 44.10 (2010), pp. 1220–1241.

[9] Petter Næss, Morten Skou Nicolaisen, and Arvid Strand. "Traffic forecasts ignoring induced demand: a shaky fundament for cost-benefit analyses." In: *European Journal of Transport and Infrastructure Research* 12.3 (2012).

[10] Susan A Shaheen, Daniel Sperling, and Conrad Wagner. "A Short History of Carsharing in the 90's." In: (1999).

[11] Susan Shaheen, Daniel Sperling, and Conrad Wagner. "Carsharing in Europe and North American: past, present, and future." In: (1998).

[12] NT Fellows and DE Pitfield. “An economic and operational evaluation of urban car-sharing.” In: *Transportation Research Part D: Transport and Environment* 5.1 (2000), pp. 1–10.

[13] Ulrike Huwer. “Public transport and csar-sharing—benefits and effects of combined services.” In: *Transport Policy* 11.1 (2004), pp. 77–87.

[14] Patrick Jochem et al. “Does free-floating carsharing reduce private vehicle ownership? The case of SHARE NOW in European cities.” In: *Transportation Research Part A: Policy and Practice* 141 (2020), pp. 373–395.

[15] T Oxley et al. “Pollution abatement from road transport: cross-sectoral implications, climate co-benefits and behavioural change.” In: *Environmental Science & Policy* 19 (2012), pp. 16–32.

[16] Xianqiang Mao et al. “Achieving CO2 emission reduction and the co-benefits of local air pollution abatement in the transportation sector of China.” In: *Environmental science & policy* 21 (2012), pp. 1–13.

[17] A. Ciociola et al. “UMAP: Urban mobility analysis platform to harvest car sharing data.” In: *proceedings of the IEEE Conference on SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI).* Aug. 2017, pp. 1–8. DOI: 10.1109/UIC-ATC.2017.8397566.

[18] *car2go API.* https://github.com/car2go/openAPI. Accessed: 10/03/2017.

[19] Sascha Herrmann, Frederik Schulte, and Stefan Voß. “Increasing acceptance of free-floating car sharing systems using smart relocation strategies: a survey based study of car2go Hamburg.” In: *International Conference on Computational Logistics.* Springer. 2014, pp. 151–162.

[20] Frederik Schulte and Stefan Voß. “Decision support for environmental-friendly vehicle relocations in free-floating car sharing systems: The case of car2go.” In: *Procedia CIRP* 30 (2015), pp. 275–280.

[21] S. Wagner, T. Brandt, and D. Neumann. “Data Analytics in Free-Floating Carsharing: Evidence from the City of Berlin.” In: *2015 48th Hawaii International Conference on System Sciences.* Jan. 2015, pp. 897–907. DOI: 10.1109/HICSS.2015.112.

[22] Stefan Schmöller et al. “Empirical analysis of free-floating carsharing usage: The Munich and Berlin case.” In: *Transportation Research Part C: Emerging Technologies* 56 (2015), pp. 34–51. ISSN: 0968-090X. DOI: http://dx.doi.org/10.1016/j.trc.2015.03.008. URL: http://www.sciencedirect.com/science/article/pii/S0968090X1500087X.

[23] Johanna Kopp, Regine Gerike, and Kay Axhausen. "Do sharing people behave differently? An empirical evaluation of the distinctive mobility patterns of free-floating car-sharing members." In: *Transportation* 42.3 (2015), pp. 449–469. URL: http://EconPapers.repec.org/RePEc:kap:transp:v:42:y:2015:i:3:p:449-469.

[24] Jörg Firnkorn. "Triangulation of two methods measuring the impacts of a free-floating carsharing system in Germany." In: *Transportation Research Part A: Policy and Practice* 46.10 (2012), pp. 1654–1672. ISSN: 0965-8564. DOI: http://dx.doi.org/10.1016/j.tra.2012.08.003. URL: http://www.sciencedirect.com/science/article/pii/S0965856412001334.

[25] F. Ciari, N. Schuessler, and K. W. Axhausen. "Estimation of carsharing demand using an activity-based microsimulation approach: model discussion and some results." In: *International Journal of Sustainable Transportation* 7.1 (2013), pp. 70–84.

[26] J. Tyndall. "Where No Cars Go: Free-Floating Carshare and Inequality of Access." In: *International Journal of Sustainable Transportation* just-accepted (2016).

[27] Jörg Firnkorn and Martin Müller. "What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm." In: *Ecological Economics* 70.8 (2011), pp. 1519–1528. ISSN: 0921-8009. DOI: http://dx.doi.org/10.1016/j.ecolecon.2011.03.014. URL: http://www.sciencedirect.com/science/article/pii/S0921800911001030.

[28] X. Wang, D. MacKenzie, and Z. Cui. *Complement or Competitior? Comparing car2go and Transit Travel Times, Prices, and Usage Patterns in Seattle.* Tech. rep. 2017.

[29] K. Kortum, B. Stolte R. Schönduwe, and B. Bock. "Free-Floating Carsharing: City-Specific Growth Rates and Success Factors." In: *Transportation Research Procedia* 19 (2016), pp. 328–340.

[30] F. Ferrero et al. "Business models and tariff simulation in car-sharing services." In: (2016).

[31] Victor A. Alencar et al. "Characterizing client usage patterns and service demand for car-sharing systems." In: *Information Systems* (2019), p. 101448. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2019.101448. URL: http://www.sciencedirect.com/science/article/pii/S0306437919305009.

[32] Juan C Herrera et al. "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment." In: *Transportation Research Part C: Emerging Technologies* 18.4 (2010), pp. 568–583.

[33] S. Ma, Y. Zheng, and O. Wolfson. "T-share: A large-scale dynamic taxi ridesharing service." In: *proceedings of the IEEE Conference on Data Engineering (ICDE), 29th International Conference*. Apr. 2013, pp. 410–421. DOI: 10.1109/ICDE.2013.6544843.

[34] Chiara Boldrini, Raffaele Bruno, and Marco Conti. "Characterising demand and usage patterns in a large station-based car sharing system." In: *proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2016, pp. 572–577.

[35] Henrik Becker, Francesco Ciari, and Kay W Axhausen. "Comparing car-sharing schemes in Switzerland: User groups and usage patterns." In: *Transportation Research Part A: Policy and Practice* 97 (2017), pp. 17–29.

[36] Susan A. Shaheen. "Mobility and the sharing economy." In: *Transport Policy* 51.Supplement C (2016), pp. 141–142. ISSN: 0967-070X. DOI: https://doi.org/10.1016/j.tranpol.2016.01.008. URL: http://www.sciencedirect.com/science/article/pii/S0967070X16000020.

[37] Francesco Ciari, Benno Bock, and Michael Balmer. "Modeling station-based and free-floating carsharing demand: test case study for Berlin." In: *Transportation Research Record: Journal of the Transportation Research Board* 2416 (2014), pp. 37–47.

[38] Elliot Martin and Susan Shaheen. "The impact of carsharing on public transit and non-motorized travel: an exploration of North American carsharing survey data." In: *Energies* 4.11 (2011), pp. 2094–2114.

[39] Tai Stillwater, Patricia Mokhtarian, and Susan Shaheen. "Carsharing and the built environment: Geographic information system-based study of one US operator." In: *Transportation Research Record: Journal of the Transportation Research Board* 2110 (2009), pp. 27–34.

[40] Jon Burkhardt and Adam Millard-Ball. "Who is attracted to carsharing?" In: *Transportation Research Record: Journal of the Transportation Research Board* 1986 (2006), pp. 98–105.

[41] Robert Cervero and Yuhsin Tsai. "City CarShare in San Francisco, California: second-year travel demand and car ownership impacts." In: *Transportation Research Record: Journal of the Transportation Research Board* 1887 (2004), pp. 117–127.

[42] Susan A Shaheen et al. *Carsharing and public parking policies: assessing benefits, costs, and best practices in North America*. Tech. rep. 2010.

[43] M. Cocca et al. "Free Floating Electric Car Sharing: A Data Driven Approach for System Design." In: *IEEE Transactions on Intelligent Transportation Systems* (2019), pp. 1–13. DOI: 10.1109/TITS.2019.2932809.

[44] Felipe Rooke et al. "Caracterização de Padrões de Demanda e Uso de um Sistema de Compartilhamento de Veículos de Duas Vias." In: *Workshop de Computação Urbana COURB-SBRC* 2.1/2018 (2018). ISSN: 2595-2706. URL: http://portaldeconteudo.sbc.org.br/index.php/courb/article/view/2341.

[45] Felipe Rooke et al. "Characterizing Usage Patterns and Service Demand of a Two-Way Car-Sharing System." In: *Big Social Data and Urban Computing*. Springer Nature Switzerland AG, 2019, pp. 01–15.

[46] Michele Cocca et al. "Free floating electric car sharing design: Data driven optimisation." In: *Pervasive and Mobile Computing* 55 (2019), pp. 59–75. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2019.02.007. URL: http://www.sciencedirect.com/science/article/pii/S1574119218305170.

[47] Malte F. Jung et al. "Displayed Uncertainty Improves Driving Experience and Behavior: The Case of Range Anxiety in an Electric Car." In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015.

[48] Michele Cocca et al. "Free Floating Electric Car Sharing in Smart Cities: Data Driven System Dimensioning." In: *4th IEEE International Conference on Smart Computing*. 2018.

[49] P. Phonrattanasak and N. Leeprechanon. "Optimal placement of EV fast charging stations considering the impact on electrical distribution and traffic condition." In: *2014 Conference and Utility Exhibition on Green Energy for Sustainable Development (ICUE)*. 2014.

[50] T.D. Chen, K.M. Kockelman, and M Khan. "The electric vehicle charging station location problem: A parking-based assignment method for Seattle." In: *92nd Transportation Research Board Meeting*. 2013, pp. 13–1254.

[51] Zihao Jiao et al. "Data-Driven Approach to Operation and Location Considering Range Anxiety of One-Way Electric Vehicles Sharing System." In: *8th International Conference on Applied Energy*. 2016.

[52] Jorg Firnkorn and Martin Muller. "Free-floating electric carsharing-fleets in smart cities: The dawning of a post-private car era in urban environments?" In: *Environmental Science & Policy* 45 (2015), pp. 30–40. ISSN: 1462-9011. DOI: https://doi.org/10.1016/j.envsci.2014.09.005.

[53] Simone Weikl and Klaus Bogenberger. "A practice-ready relocation model for free-floating carsharing systems with electric vehicles – Mesoscopic approach and field trial results." In: *Transportation Research Part C: Emerging Technologies* 57 (2015), pp. 206–223. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2015.06.024.

[54] Andrea Hess et al. "Optimal Deployment of Charging Stations for Electric Vehicular Networks." In: *Proceedings of the First Workshop on Urban Networking*. Nice, France, 2012, pp. 1–6. ISBN: 978-1-4503-1781-8.

[55] Jörg Firnkorn and Martin Müller. "Selling Mobility instead of Cars: New Business Strategies of Automakers and the Impact on Private Vehicle Holding." In: *Business Strategy and the Environment* 21.4 (2012), pp. 264–280. ISSN: 1099-0836. DOI: 10.1002/bse.738.

[56] Shiva Habibi et al. "Comparison of free-floating car sharing services in cities." In: *European Council of Energy Efficient Economy Summer Study*. 2016, pp. 771–778.

[57] Katherine Kortum et al. "Free-Floating Carsharing: City-Specific Growth Rates and Success Factors." In: *Transportation Research Procedia* 19 (2016), pp. 328–340.

[58] Michele Cocca et al. "A Data Driven Optimization of Charging Station Placement for EV Free Floating Car Sharing." In: *21st IEEE International Conference on Intelligent Transportation Systems*. 2018.

[59] Maria Carmen Falvo et al. "EV charging stations and modes: International standards." In: *2014 Symposium on Power Electronics, Electrical Drives, Automation and Motion*. 2014. DOI: 10.1109/SPEEDAM.2014.6872107.

[60] T.A.A. Rickenberg, A Gebhardt, and M.H. Breitner. "A decision support system for the optimization of car sharing stations." In: *ECIS 2013 - Proceedings of the 21st European Conference on Information Systems* (Jan. 2013).

[61] Sonneberg Marc, Kuehne Kathrin, and Michaelm Breitner. "A Decision Support System for the Optimization of Electric Car Sharing Stations." In: 2015.

[62] Christoph Flath, Jens Ilg, and Christof Weinhardt. "Decision Support for Electric Vehicle Charging." In: *Proceedings of the 18th Americas Conference on Information Systems (AMCIS 2012), Seattle, Washington, USA, 9 - 12 August 2012. Association for Information Systems*. 2012.

[63] Alfred Brendel et al. "A Decision Support System for Computation of Carsharing Pricing Areas and its Influence on Vehicle Distribution." In:

[64] Alfred Brendel et al. "Adapting Carsharing Vehicle Relocation Strategies for Shared Autonomous Electric Vehicle Services." In: 2017.

[65] Alfred Benedikt Brendel, Christian Rockenkamm, and Lutz Maria Kolbe. "Generating Rental Data for Car Sharing Relocation Simulations on the Example of Station-Based One-Way Car Sharing." In: *HICSS*. 2017.

[66] Sebastian Wagner et al. "Data Analytics for Location-Based Services: Enabling User-Based Relocation of Carsharing Vehicles." In: *ICIS*. 2015.

[67] Ran Bi et al. "A simulation-based heuristic for city-scale electric vehicle charging station placement." In: *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. 2017, pp. 1–7.

[68] Matthias Eisel et al. "Applying Demand Response Programs for Electric Vehicle Fleets." In: *AMCIS*. 2015.

[69] Singiresu S. Rao. *Engineering Optimization: Theory and Practice: Fourth Edition*. English. John Wiley and Sons, 2009. ISBN: 9780470183526.

[70] Christodoulos C. A. Floudas and P. M. Pardalos. *Encyclopedia of Optimization*. Springer-Verlag, 2006. ISBN: 0387336249.

[71] Stephen J. Wright. "Coordinate Descent Algorithms." In: *Math. Program.* 151.1 (June 2015), pp. 3–34. ISSN: 0025-5610.

[72] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Addison-Wesley Longman Publishing, 1989. ISBN: 0201157675.

[73] M. Smith and J. Castellano. *Costs Associated With Non-Residential Electric Vehicle Supply Equipment: Factors to consider in the implementation of electric vehicle charging stations*. Tech. rep. 2015.

[74] Michele Cocca et al. "On Car-Sharing Usage Prediction with Open Socio-Demographic Data." In: *Electronics* 9.1 (2020), p. 72.

[75] Todd Alexander Litman. "Evaluating Carsharing Benefits." In: *GeoJournal* 1702 (2015).

[76] Francesco Ciari, Claude Weis, and Milos Balac. "Evaluating the influence of carsharing stations' location on potential membership: a Swiss case study." In: *EURO Journal on Transportation and Logistics* 5.3 (2016), pp. 345–369.

[77] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.

[78] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[79] Iwao Okutani and Yorgos J Stephanedes. "Dynamic prediction of traffic volume through Kalman filtering theory." In: *Transportation Research Part B: Methodological* 18.1 (1984), pp. 1–11.

[80] Stephen Clark. "Traffic prediction using multivariate nonparametric regression." In: *Journal of transportation engineering* 129.2 (2003), pp. 161–168.

[81] Mario Catalano, Barbara Lo Casto, and Marco Migliore. "Car sharing demand estimation and urban transport demand modelling using stated preference techniques." In: *European Transport* 40 (2008), pp. 33–50. ISSN: 1825-3997.

[82] Stefan Schmöller and Klaus Bogenberger. "Analyzing external factors on the spatial and temporal demand of car sharing systems." In: *Procedia-Social and Behavioral Sciences* 111 (2014), pp. 8–17.

[83] Suining He and Kang G. Shin. “Spatio-temporal Adaptive Pricing for Balancing Mobility-on-Demand Networks.” In: *ACM Trans. Intell. Syst. Technol.* 10.4 (2019), 39:1–39:28. ISSN: 2157-6904. DOI: 10.1145/3331450. URL: http://doi.acm.org/10.1145/3331450.

[84] Pierre Hulot, Daniel Aloise, and Sanjay Dominik Jena. “Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems.” In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*. KDD ’18. London, United Kingdom: ACM, 2018, pp. 378–386. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219873. URL: http://doi.acm.org/10.1145/3219819.3219873.

[85] Dong Wang et al. “DeepSD: supply-demand prediction for online car-hailing services using deep neural networks.” In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE. 2017, pp. 243–254.

[86] Stefan Schmöller et al. “Empirical analysis of free-floating carsharing usage: The Munich and Berlin case.” In: *Transportation Research Part C: Emerging Technologies* 56 (2015), pp. 34–51.

[87] Raj Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.

[88] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[89] Michelangelo Barulli et al. “On Scalability of Electric Car Sharing in Smart Cities.” In: *2020 IEEE International Smart Cities Conference (ISC2)*. IEEE, pp. 1–8.

[90] Jennifer L Kent. “Carsharing as active transport: What are the potential health benefits?” In: *Journal of Transport & Health* 1.1 (2014), pp. 54–62.

[91] Elliot W Martin and Susan A Shaheen. “Greenhouse gas emission impacts of carsharing in North America.” In: *IEEE Transactions on intelligent transportation systems* 12.4 (2011), pp. 1074–1086.

[92] Todd Litman. “Evaluating carsharing benefits.” In: *Transportation Research Record* 1702.1 (2000), pp. 31–35.

[93] Caryn Green. *Car-Sharing – Good for the Environment and the Budget*. Accessed: 2020-07-09. 2009.

[94] Mario Rossi. “Mobilità alternativa Crack sharing.” In: *Quattroruote* 746 (2017). URL: http://shoped.it/shop/prodotto/quattroruote-ottobre-2017/.

[95] Pier Luigi del Viscovo. *Car sharing, in crescita +30% all’anno fino al 2021*. Accessed: 2020-07-09. 2019.

[96] Alessandro Ciociola et al. “UMAP: Urban Mobility Analysis Platform to Harvest Car Sharing Data.” In: *Proceedings of the IEEE Smart City Innovations*. IEEE. 2017.

[97] Alessandro Ciociola et al. "Impact of Charging Infrastructure and Policies on Electric Car Sharing Systems." In: *Proceedings of the 23rd IEEE International Conference on Intelligent Transportation Systems.* IEEE. 2020.

[98] Rahul Nair and Elise Miller-Hooks. "Fleet management for vehicle sharing operations." In: *Transportation Science* 45.4 (2011), pp. 524–540.

[99] Long He et al. "Service region design for urban electric vehicle sharing systems." In: *Manufacturing & Service Operations Management* 19.2 (2017), pp. 309–327.

[100] Min Xu and Qiang Meng. "Fleet sizing for one-way electric carsharing services considering dynamic vehicle relocation and nonlinear charging profile." In: *Transportation Research Part B: Methodological* 128 (2019), pp. 23–49.

[101] Francesco Ciari, Michael Balmer, and Kay W Axhausen. "Concepts for a large scale car-sharing system: Modelling and evaluation with an agent-based approach." In: *Working Paper/IVT* 517 (2008).

[102] Francesco Ciari, Milos Balac, and Kay W Axhausen. "Modeling carsharing with the agent-based simulation MATSim: State of the art, applications, and future developments." In: *Transportation Research Record* 2564.1 (2016), pp. 14–20.

[103] Shiva Habibi et al. "Comparison of free-floating car sharing services in cities." In: *Proceedings of the ECEEE Summer Study.* 2017.

[104] Remi Tachet et al. "Scaling law of urban ride sharing." In: *Scientific reports* 7 (2017), p. 42868.

[105] Xiaopeng Li et al. "Design framework of large-scale one-way electric vehicle sharing systems: A continuum approximation model." In: *Transportation Research Part B: Methodological* 88 (2016), pp. 21–45.

[106] Michele Cocca et al. "Free floating electric car sharing design: Data driven optimisation." In: *Pervasive and Mobile Computing* 55 (2019), pp. 59–75.

[107] Michele Cocca et al. "Free Floating Electric Car Sharing: A Data Driven Approach for System Design." In: *IEEE Transactions on Intelligent Transportation Systems* 20.12 (2019), pp. 4691–4703.

[108] Yikang Hua et al. "Joint infrastructure planning and fleet management for one-way electric car sharing under time-varying uncertain demand." In: *Transportation Research Part B: Methodological* 128 (2019), pp. 185–206.

[109] Rafael FF Lemme, Edilson F Arruda, and Laura Bahiense. "Optimization model to assess electric vehicles as an alternative for fleet composition in station-based car sharing systems." In: *Transportation Research Part D: Transport and Environment* 67 (2019), pp. 173–196.

[110] Martinez Vasconcelos, Guiames Correia, and Farias. “Environmental and financial impacts of adopting alternative vehicle technologies and relocation strategies in station-based one-way carsharing: An application in the city of Lisbon, Portugal.” In: *Transportation Research Part D: Transport and Environment* 57 (2017), pp. 350–362. ISSN: 1361-9209.

[111] Aditya Krishna Menon and Young Lee. “Predicting Short-Term Public Transport Demand via Inhomogeneous Poisson Processes.” In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 2207–2210. ISBN: 978-1-4503-4918-5. DOI: 10.1145/3132847.3133058.

[112] Raúl Tomás García et al. “The kernel density estimation for the visualization of spatial patterns in urban studies.” In: *15th International Multidisciplinary Scientific GeoConference SGEM 2015*. 2015, pp. 867–874. DOI: 10.5593/SGEM2015/B21/S8.111.

[113] Mercedes-Benz. *Leasing cost.* Accessed: 2020-07-09. 2020.

[114] Chris Nelder and Emily Rogers. *Reducing EV charging infrastructure costs.* Accessed: 2020-07-09. 2019.

[115] Comune di Torino. *Ground occupation tax for the city of Turin.* Accessed: 2020-07-09. 2020.

[116] Eurostat. *Electricity price statistics.* Accessed: 2020-07-09. 2019.

[117] Infodata. *Il costo del lavoro divide in due l’Europa.* Accessed: 2020-07-09. 2019.

[118] *Doctor Car Wash Prices.* Accessed: 2020-07-09. 2020.

[119] *Car2Go Prices in Turin.* Accessed: 2020-07-09. 2020.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was LuaLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi =scudo`. This class is available in every up-to-date and complete TEX-system installation.