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

Conversion of FFCS ICE fleet to EV

a data-driven approach

Michele Cocca
* * * * * *

Supervisor
Prof. Marco Mellia, Supervisor

Doctoral Examination Committee:
Prof. A.B., Referee, University of …
Prof. C.D., Referee, University of …
Prof. E.F., University of …
Prof. G.H., University of …
Prof. I.J., University of …

Politecnico di Torino
September 25, 2020

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Michele Cocca

Turin, September 25, 2020

Summary

This is where you write your abstract ... (Maximum 4000 characters, i.e. maximum two pages in normal sized font, typeset with the thesis layout).

The abstract environment is also available, but `\summary` is preferred because it generates an un-numbered chapter. The abstract environment is more suitable for articles and two column typesetting without a separate title page.

Acknowledgements

And I would like to acknowledge …

Acknowledgements are mandatory only when people outside the academic institution supported the development of the research that was performed in order to reach the conclusion of the doctorate program.

I would like to dedicate this thesis to my loving parents

The dedication very seldom is a proper thing to do; in some countries it is very common, while in other countries it is done for imitation of other people habits.

The sentence used above clearly is an example of something very common, but it is useless. Of course we all love our beloved parents, but it is not necessary to "engrave it in stone".

Contents

Chapter 1

Introduction

descrivo la storia i benefit e la storia del carsharing

Chapter 2

Data acquisition

2.1 Abstract

Free Floating Car Sharing (FFCS) is a popular business model based on the shared-economy paradigm. The users can pick and drop the cars within a given operative area paying only for the time spent driving. This system implies front-end-crew-free reservation and releasing procedure, possible through a FFCS provider's web application. It exposes the coordinates of each available car and making them bookable by only one tap.

In this chapter, I describe how continuously monitoring two FFCS providers and how I created a dataset of FFCS trips. To do that, I developed the *Urban Mobility Analysis Platform* (*UMAP*) able to fetches data from car sharing platforms in real time. Secondly, *UMAP* processes the data to extract advanced information about driving patterns and user's habits. To extract information, *UMAP* augments the data available from the car sharing platforms with mapping and direction information fetched from other web platforms. This information is stored in a data lake where historical series are built, and later analyzed using analytics modules easy to design and customize.

In total *UMAP* collected [mike: trips and bookinfs per c2g and engjoy for each city]

This chapter refers mostly my paper " *UMAP: Urban mobility analysis platform to harvest car sharing data* Ciociola et al., 2017, presented in at 2017 IEEE SmartWord conference

2.2 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 policy makers are positively looking into "smart" approaches to improve the current status of their urban network. The ability to collect data, is the first step to take informed decisions. Unfortunately,

getting information about mobility patterns and human driving habits is not easy because of both technical challenges and privacy issues. To this extent, in this chapter I describe the possibility of harvesting data openly exposed on the Web to obtain information about mobility habits in cities, and make it available to the players by using a smart-platform. In particular on car sharing platforms and mapping and direction services.

Car sharing refers to a model of car rental where customers rent a car for a short period of time, usually for a few hours or less. One of its most interesting systems is the so called *Free-Floating Car Sharing (FFCS)* system. The peculiarity of this system 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. Moreover, other country-based providers exists like *Enjoy* in Italy, operative in 6 cities.

This chapter describes how *UMAP* collects, processes, augments, and stores data in a data lake, make the data 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 are 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 in maintenance), or platform design (e.g., synchronous or asynchronous updates).

In total, *UMAP* collected about XXX of trips in X cities for car2go, working from 2017 to 2018. Instead, considering Enjoy, I get about YYYY rides from YY cities from 2017 to 2020 [mike: rivedere dati su quanti bookings ci sono]. The source code of *UMAP* for research purposes.[2]

The reminder of this chapter is structured as follows: Sec. 7.3 discusses the related work. Sec. 2.3 describes in details the raw data structure and the data flow from providers' API to the middle stage. Sec. 2.4 describes how the software implements the data raw elaboration to trips records and their storage into data lake . [mike: continuare con altre sezione se ne aggiungo]

2.3 Data Acquisition

In this section, I provide a description of *UMAP* structure. Figure 2.1 depicts the architecture of *UMAP*, composed by a first module for the data acquisition, by a second

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

[2] github.com/MobilityPolito/

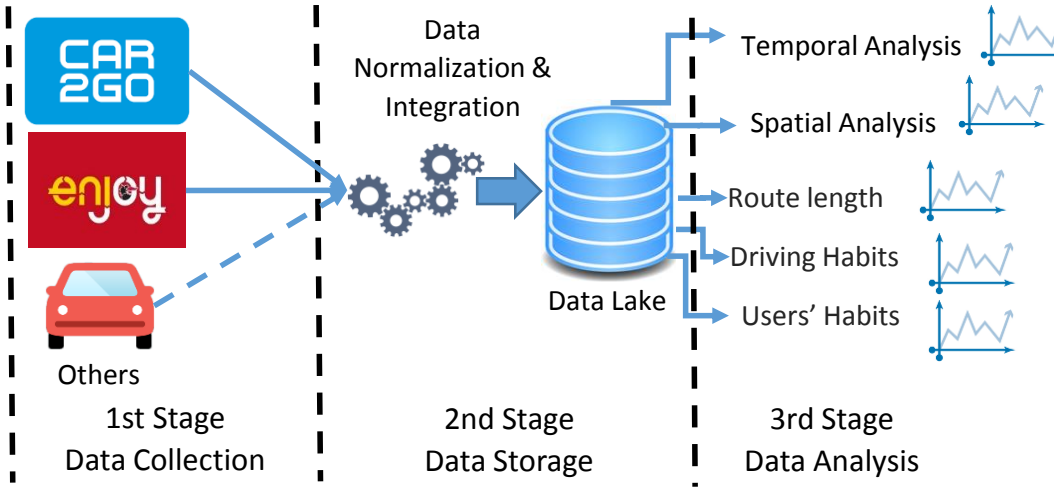


Figure 2.1: *UMAP* overview

module for data normalization and integration, and then a third module for the data analysis.

The first module consists in 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 *car2go API* n.d., 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 the list of available cars. Through this analysis, we obtain both the URL used to request the list of available cars, and how fetch the data for a specific city. Both system 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 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*, we obtain the JSON snapshot *S* listing the available cars. The sampling period has been set to one minute, to balance 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 different format. For this study, I collect each car unique identifier and current geo-location 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, we design a data integration step to have common names for fields containing the same information, if present.

2.4 Data Normalization and Integration

In this second module I illustrates hoe *UMAP* processes and consolidate each snapshot to obtain *parking* and *bookings* periods for each car. A *parking* is time where the car is available for a user ride. On the other hand a *bookings* is time elapsing 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. The framework 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. The software records this event, with 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 on *MongoDB*, a NoSQL document-based database. A MongoDB database includes a set collections, i.e., groups of documents. Each document is a set of key-value pairs, organized in a JSON structure. The schema-less structure of MongoDB fits well in our work, because it can handle in the same collection documents defined with different key-value pairs. We decide to rely on such a system as we can easily manage the different field structures of providers, car2go and Enjoy in our use case. In addition, MongoDB offers a 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

[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.

query each car last observed status, and update it. These are also instrumentals for a 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 past state of cars, for past bookings and parkings, respectively.

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

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. We will be back on this 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

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

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, we set 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. We finalize the previous parking period by setting the *FinalTime* in the *ActiveParking* collection – line 21-22. At last, we create a new booking via the *new Booking* function – line 23.

I let *UMAP* scrape car2go's data [mike: prendere le date]. In total it is possible to count about 27 million bookings spread in 23 cities. The table 2.1 reports a brief resume of all the bookings, parkings present in the data lake.

[mike: devo prendere le date anche di enjoy]

2.5 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 offer a large number of libraries that easily interact with the different technologies like GIS, maps and MongoDB. In particular the usage of GeoPandas allow me to easily perform geographic analysis and split the city in many areas (or zones) of any possible shapes. I present more detailed characterization in chapter 3.

2.6 Conclusion

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 snapshots the system getting all the car ready for a ride.

The seconds step illustrates the algorithm that compares consecutive snapshots detecting car status variation. Here I introduced the two fundamental car status I use to

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

[5] wikipedia.org

Table 2.1: Overview of car2go's data

City	City Size $[km^2]$[5]	Population [5]	Avg. Fleat	Bookings
Columbus	17	892k	187	186k
Florence	34	372k	220	333k
Denver	36	727k	312	348k
Austin	31	964k	315	377k
Frankfurt	40	701k	242	505k
Toronto	49	3120k	400	536k
Amsterdam	38	854k	314	573k
Montreal	49	1704k	429	606k
New York City	77	8522k	500	739k
Turin	47	874k	396	868k
Munich	61	1464k	478	916k
Washington DC	64	705k	563	919k
Stuttgart	59	632k	486	1001k
Seattle	92	744k	710	1134k
Calgary	47	1239k	552	1176k
Rome	65	2837k	582	1240k
Rheinland	82	1688k	648	1421k
Vienna	72	1915k	688	1702k
Madrid	42	3233k	424	2092k
Milan	77	1396k	776	2223k
Hamburg	82	1833k	812	2561k
Vancouver	78	631k	977	2701k
Berlin	125	3769k	1009	3091k

describe each car history: *parkings* and *bookings*.

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

Chapter 3

Dataset Characterization

3.1 Abstract

Car sharing is nowadays a popular means of transport in smart cities. In particular, the free-floating paradigm lets the customers look for available cars, reserve one, and then start and stop the rental at their will, within a specific area. This is done thanks to a smartphone app, which contacts a web-based back-end to exchange information. This kind of service relies on a well structured IT infrastructure, that may expose data on the providers' fleet.

In this chapter, I prove the flexibility of *UMAP*, presented in chapter 2, considering as a case of study the city of Turin (Italy). I collect FFCS vehicles usage data for over 50 days to characterize both the temporal and spatial properties of rentals, and to characterize customers' habits in using the service, compared to public transportation alternatives. Results provide insights about the driving style and needs, which are useful for smart city planners, and prove the feasibility of our approach.

This work refers mostly my paper " *UMAP: Urban mobility analysis platform to harvest car sharing data*, presented in at 2017 IEEE SmartWord conference Ciociola et al., 2017

3.2 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 policy makers 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 take informed decisions. Unfortunately, getting information about mobility patterns and human driving habits is not easy because of both technical challenges and privacy issues.

Car sharing refers to a model of car rental where customers rent a car for a short period of time, usually for a few hours or less. One of its most interesting systems is

the so called *Free-Floating Car Sharing (FFCS)* system. The peculiarity of this system 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. Then, with a simple tap they can book a car, and start/end the rental. The FFCS app contacts a web-based back-end server to fetch data about available cars, perform a booking, and accounting operations. Typically for this purpose web API are used, some of which are publicly documented *car2go API* n.d. 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 for free this information. In the past, this approach has been successfully used to obtain data for specific mobility studies – see Sec. 3.3 for more details. This chapter relies on the tool described in chapter 2. In particular, I analyze the first chunk of data I was able to collect. 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 actually made a ride, I characterize the FFCS service utilization with the aim 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 the software can collect in real time for each rental. This 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 the reservation time and the time to look for a parking spot, this may suggest that the time-based tariffs adopted by FFCS systems may encourage fast driving styles in the hope to reduce the rental cost. I next compare the duration of the booking with the equivalent trip duration by public transport as returned again 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 represents an important support tool for the investigation of car sharing users' habits. The scalable design of *UMAP* allows the policy maker to collect data from many FFCS providers and integrate it with other sources. This eases the analysis when taking in consideration trends and providers comparison. *UMAP* allows the

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

Transportation Authority to take informed decisions when planning public transport systems. This characteristic strengthens the potentiality of *UMAP* for economical 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 services improvements, both for car sharing and public transport systems. The source code of *UMAP* is available for research purposes.[2]

The reminder of this chapter is structured as follows: Sec. 3.3 discusses the related works. Sec. **??** presents the in the following order: First, car2go and Enjoy car usage over time characterization; second, 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. Sec. **??** concludes the chapter. [mike: da rivedere]

3.3 Related Work

Since the diffusion of the new form of car sharing based on a free-floating approach, many researchers from different fields have been dedicating an increasing attention to the analysis of these systems. 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 Herrmann et al., 2014; Schulte and Voß, 2015; S. Wagner et al., 2015. On the one hand, with respect to station-based car sharing, the flexibility of the free-floating system may limit the operator's control over the drop-off zones, but on the other hand allows smarter strategies. Herrmann, Schulte and Voß Herrmann et al., 2014 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 Schulte and Voß, 2015, who introduced an approach to support the decision of vehicle relocation method to reduce costs and emissions in FFCS. Those kind of investigations may result in a very useful support for the providers. In this direction, Wagner, Brandt and Neumann S. Wagner et al., 2015, analyzed the use of car sharing in Berlin, using indicators of actractiveness of certain areas, in order to develop a methodology that is able to help in business strategies, the expansion of operative areas and to react to shifts in demand. In these works, the authors used data collected from car sharing providers, using the car2go API Herrmann et al., 2014; Schulte and Voß, 2015 or by a direct cooperation S. Wagner et al., 2015.

The study of the customers' behaviour has been addressed by different researchers F. Ciari et al., 2013; Jörg Firnkorn, 2012; Kopp et al., 2015; Schmöller et al., 2015; Tyndall, 2016. Schmöller et al. Schmöller et al., 2015 studied factors that may influence the demand of car sharing, carrying out an empirical analysis, considering FFCS in Berlin and Munich.

[2] github.com/MobilityPolito/

Kopp et al. Kopp et al., 2015 inspected the behavior of two categories of users, the members of a 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 Jörg Firnkorn, 2012, who proposed in its work a triangulation of two methods applied in the same survey, to provide more precise measurements. Another approach was proposed by Ciari et al. F. Ciari et al., 2013, 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 really accessible to the people. Tyndall Tyndall, 2016 combined data of FFCS usage in ten US cites with demographic information, studying neighbourhood infrastructures, population distribution and their mobility habits. It has been showed that benefits of FFCS are distributed unequally, with a shift on usage in favor of advantaged populations.

Eco-sustainability is another important asset for car sharing services. Firnkorn and Müller Jörg Firnkorn and Müller, 2011 studied the environmental effects of FFCS in Ulm, registering lower pollution levels and a reduction of private vehicle ownership.

The goal of our work is to address all these challenges from the local administration's perspective, in order to develop new transport and mobility policies. A study of this kind was recently conducted by Wang et al. X. Wang et al., 2017 for the city of Seattle, where car2go was compared with public transport service. Kortum et al. Kortum et al., 2016 remark the necessity of use 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 to demographic informations, offered an aggregated point of view, over different cities, of the growth of the car sharing service and an understanding of the main characteristics. To the best of our knowledge, in the context of our case of study, the only work on free-floating car sharing was conducted by Ferrero et al. Ferrero et al., 2016 from an economical point of view.

The majority of the previous works Herrmann et al., 2014; Kopp et al., 2015; Kortum et al., 2016; Schmöller et al., 2015; Schulte and Voß, 2015; S. Wagner et al., 2015; X. Wang et al., 2017 leverage data collected in real-time or using surveys and interviews. Thanks to car2go APIs, which easily make avaiable car sharing data, a more data-driven approach is attractive for many researchers that start facing the problem of FFCS mobility analysis. Remarkably, only Kortum et al., 2016 seems to use data collected actively by different car sharing providers. While authors use information only for a specific purpose i.e., analyzing the trend of car sharing through the years, here we 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 in the analysis of the mobility, harvesting data collected from FFCS platforms, but also from other online systems, like mapping and direction services.

3.4 Temporal Analyses

In this section I show different analysis to discover and characterize how the FFCS are used. In the first part of the section, I analyse the temporal systems characterization 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 by 394 cars, and the fleet of Enjoy was composed by 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 defined in the chapter 2, section 2.4.

Definition 1. *the **Parking** is the time period in which the car is present in, at least, two consecutive snapshot. Therefore that car it is available for an user reservation*

Definition 2. *the **Booking** is the time period in which the car is NOT present in, at least, two consecutive snapshot. There fore an user booked that vehicle or the provider temporary removed it for maintenance.*

3.4.1 System Utilization

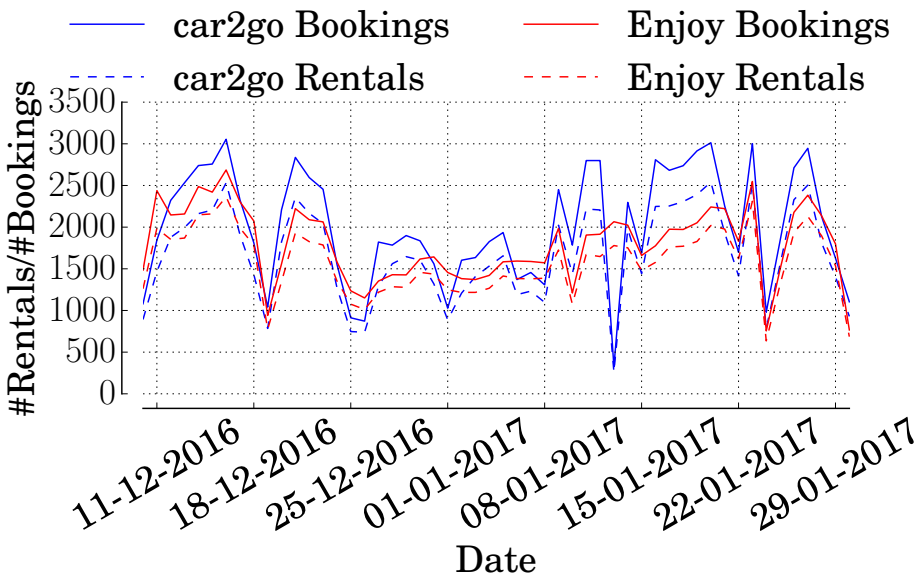


Figure 3.1: Total number of bookings and of rentals per day for car2go and for Enjoy

The providers, in this case study, allows the users to *reserve* a car before the ride. More in details, 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 it 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 the define:

Definition 3. ***Reservation*** *A reservation it 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 behaviour with the number of bookings and rentals decreasing in the Christmas period and increasing again after the Epiphany.

Secondly, despite car2go fleet has more than twice as much cars than Enjoy (394 vs. 172), the number of car2go bookings does not show such a higher value with respect to Enjoy. With Enjoy having more bookings in some snapshots e.g., December 10th and 11th. Moreover, in some points (ecember 19th, January 24th) it is possible to detect huge drop due to the failure of our crawler.

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).

Looking at the data with a 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 to 60%, versus 30% of car2go.

Even in absolute number of rentals, Enjoy shows an higher number of bookings after 8 pm during the weekdays, and always during the weekends. This can be explained by

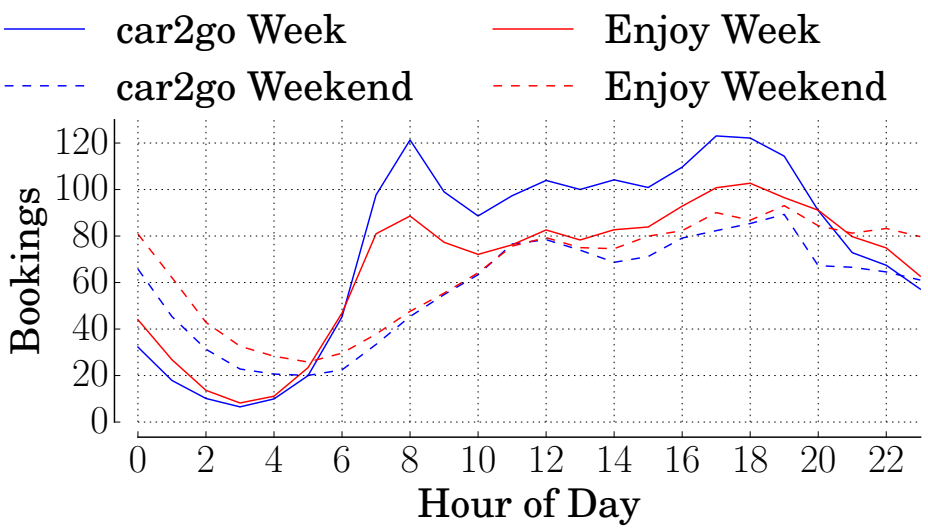


Figure 3.2: Mean number of bookings in weekdays and weekends for car2go and Enjoy

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 4 seats 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.

3.5 Rides Characterization

In this section, I give a detailed look about driving habits. In particular I compare driving distances and duration of 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 other one.

3.5.1 Driving patterns

Now, 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)

First, I check if and for how long users reserve a car and then they cancel a booking. Interestingly, only a small subset of Enjoy bookings are affected by cancellation with

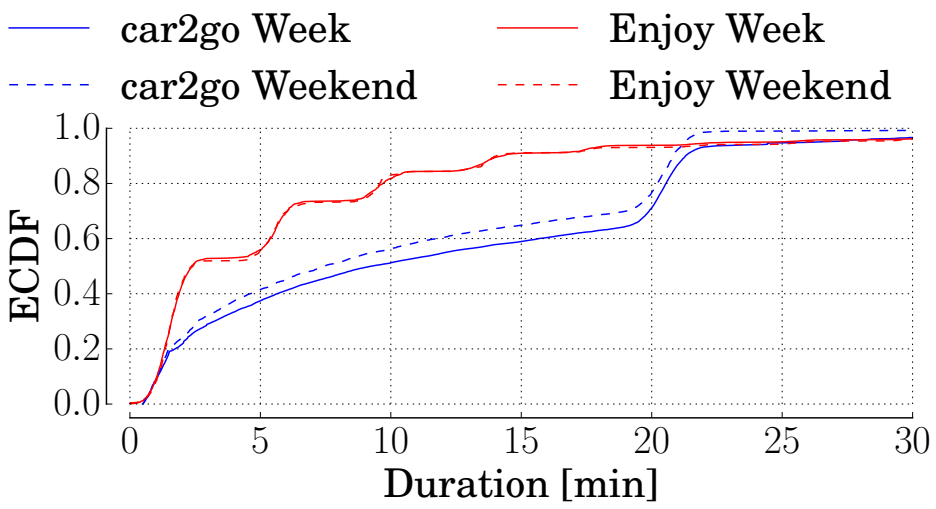


Figure 3.3: ECDF of the booking duration when the booking does not produce a rental. Weekdays and weekends

respect to car2go bookings. In particular, the dataset presents 14.9% of car2go and 2.9% of Enjoy bookings cancellation. This again hints for 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 car2go availability is higher and so it looks easier to find a closer car. People may thus cancel a previous booking when they find 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 huge 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. Truly, users, may reserve a car 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 cancellation are already done. One last important aspect that this picture shows is how the Enjoy curves have some steps instead of being smooth as the car2go ones. This hints to periodic updates on web system so that a time granularity emerges. To shed some lights on this phenomenon, I performed some active experiments with the Enjoy web portal. The experiment consists of making a new reservation and find when our crawler detects that the car actually disappears from the set of available cars. 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 we 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 of 2.5 minutes, i.e., the average delay the Enjoy system adds.

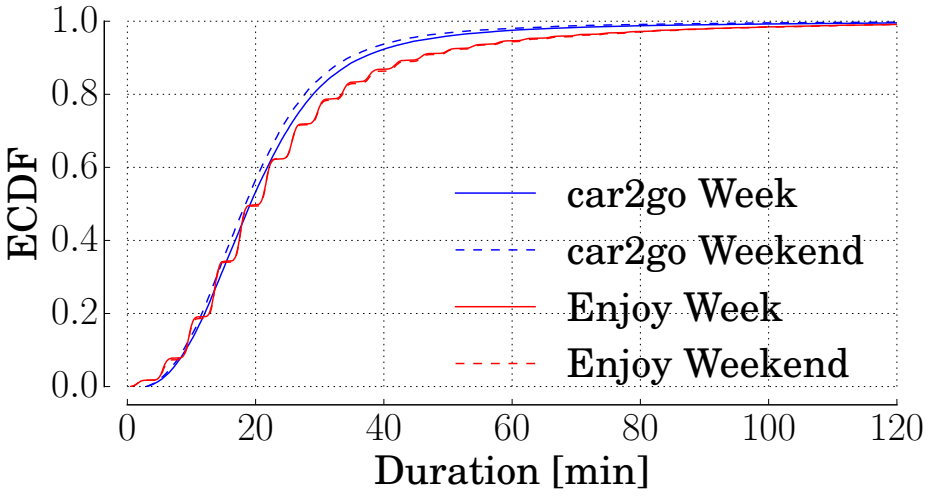


Figure 3.4: ECDF of the rental duration. Weekdays and weekends

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 then 30 minutes. It is important to remark that this times include also 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.

I repeat the same analysis considering the driving distance as reported in Figure 3.5. To determine the driving distance of each trip we 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 a comparable behaviour, and marginal changes during weekdays and weekends. Interestingly, the graphs points out that the 90% of the trips last less then 5 km demonstrating that most of the rentals are used for short trips both in term of time, and in term 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.

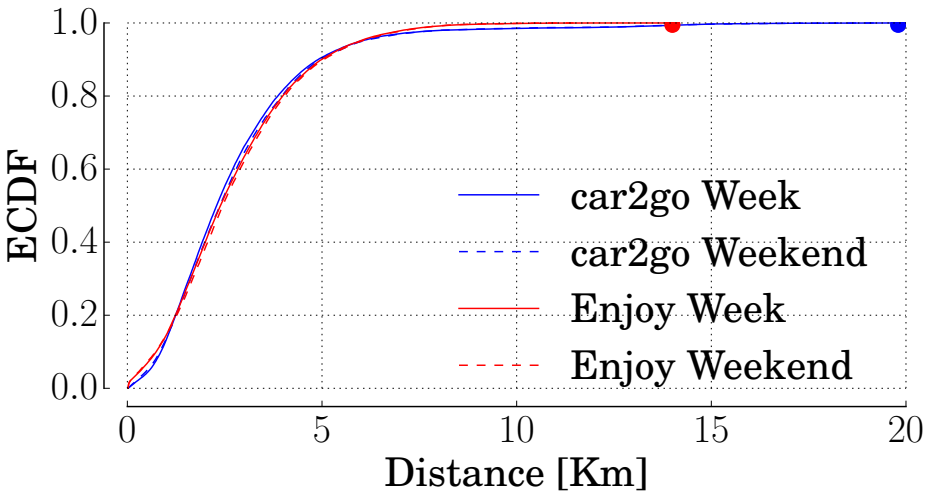


Figure 3.5: ECDF of the rental driven distance. Weekdays and weekends

3.5.2 Spatial Analysis

In the previous section I analysed 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 neighbour map. Then I computed the attractiveness of each neighbour. 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 the office hours, while the trips are sparsely ending in the suburbs during the evening.

3.5.3 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 rental initial position to the rental final 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 3.7 shows an 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 the ease of representation the values are rounded by minute. The

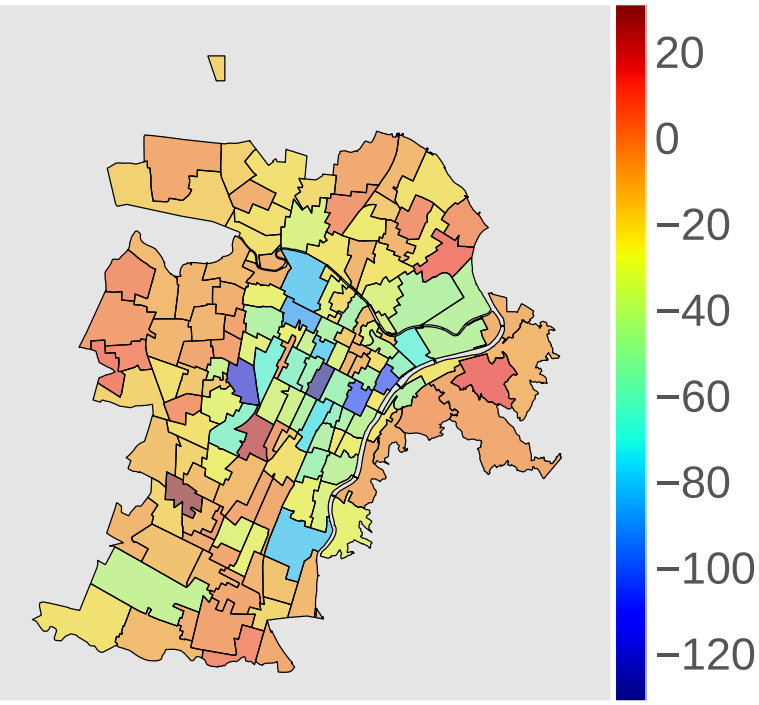


Figure 3.6: Heatmap of arrival - departure per area from 7 am to 12 am vs from 5 pm to 9 pm

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 then the driving time. However, a non negligible number of trips (12.1%) falls in the area where the booking time lasts less than the driving time.

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 3.8. It is possible to see that te most of these trips are only 5 minutes faster than the estimated driving time, with Enjoy users which seems 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 by the booking time, may have some drawbacks as it may encourage users to drive fast. An hybrid

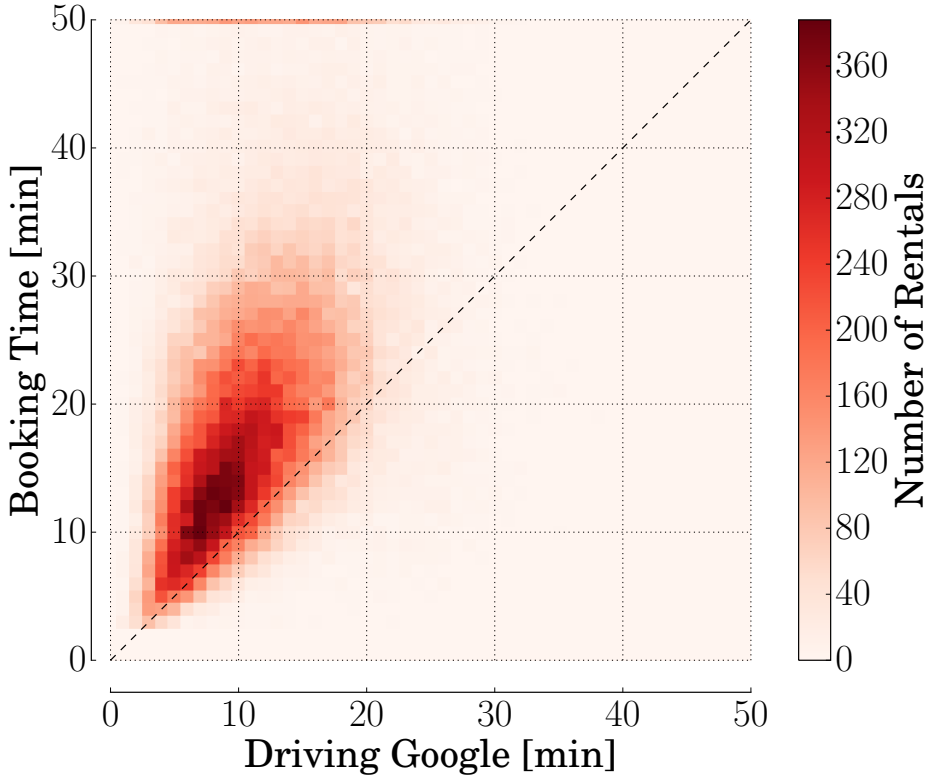


Figure 3.7: Heatmap of booking time vs estimated driving time by Google

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 an user drive faster than expected, or by reducing the fee in case of traffic congestion.

At last, I leverage Google Directions APIs to extract public transport travel information for each vehicle's trip. I want to analyse 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 the 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 in reaching the car or finding a parking spot for the drop-off.

To help to visualize the juxtaposition of car sharing and public transport, we extract from the data the probability of booking a car, conditioned to the public transport travel time. Figure **??**, 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.

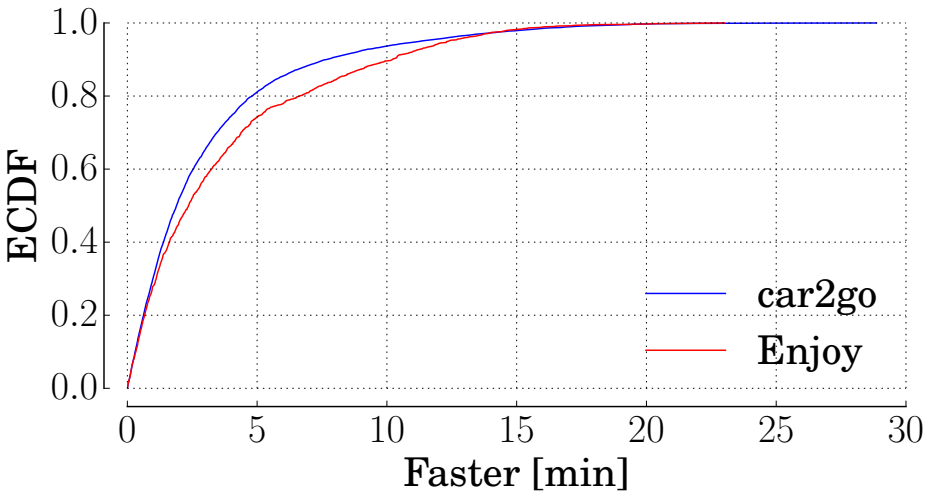


Figure 3.8: ECDF of the difference between the expected driving time and the actual driving time

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 duration greater than 30 minutes. 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 we report in Figure 3.11 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 we take all the rentals of interest, and then we compute the average value and report it. A first interesting aspect is that the average time of Enjoy is always greater then 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 ascribe this trend with traffic jam, instead, but more likely with an increase 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 then 20 minutes by car. Instead, during the daytime the average public transport time get close to the car sharing time.

3.6 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.

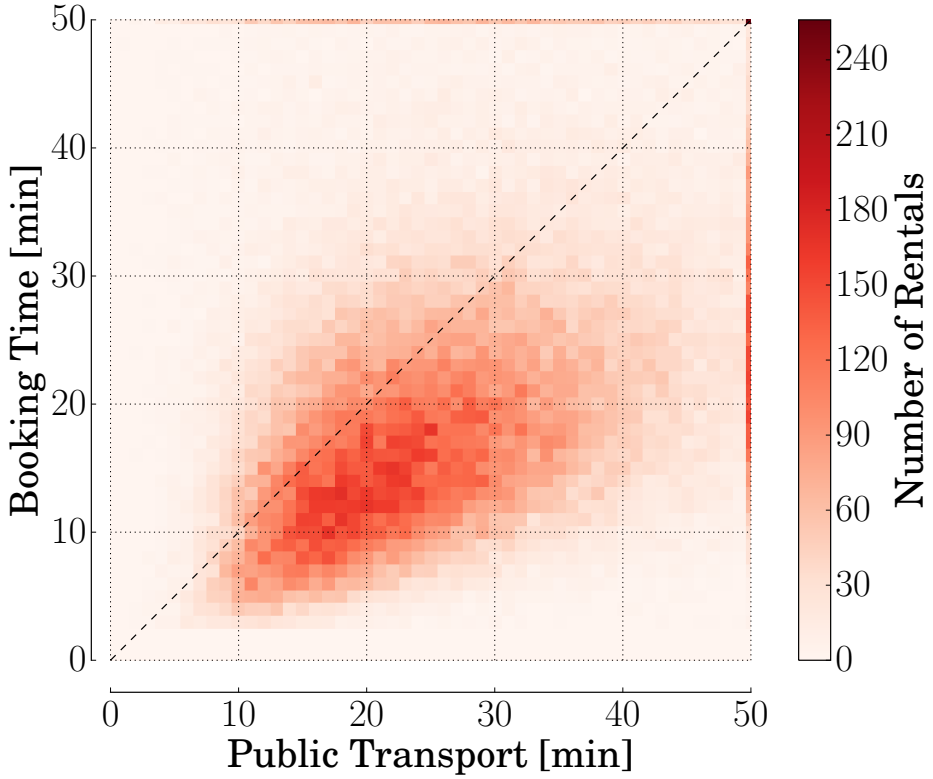


Figure 3.9: heatmap of the booking time vs the estimated public transport time by Google

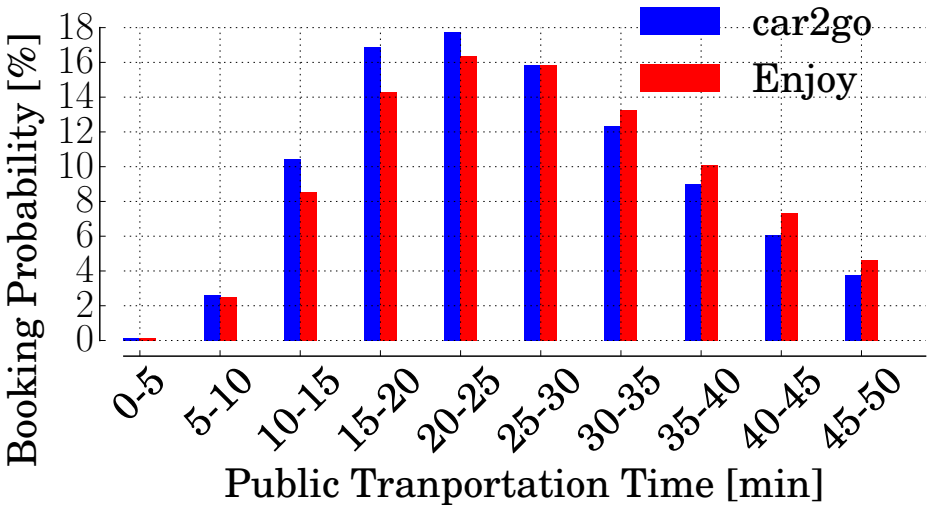


Figure 3.10: Public transportation vs car sharing

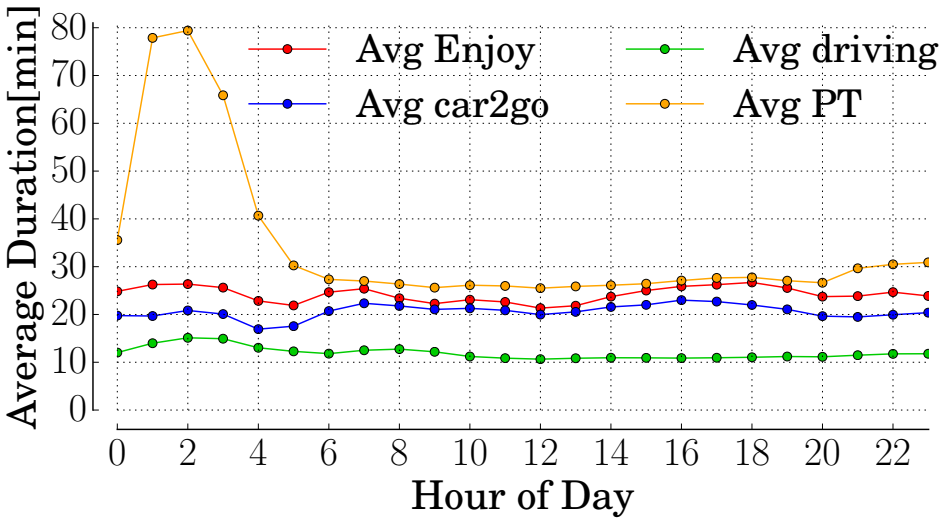


Figure 3.11: Average Time per transport solution per hour

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 big amount of information that it is possible to extract from well designed data collection pipelines. More in details, by looking the 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

4.1 Abstract

The understanding of the mobility on urban spaces is useful for the creation of smarter and sustainable cities. However, getting data about urban mobility is challenging, since only a few companies have access to accurate and updated data, that is also privacy-sensitive.

In this work, we characterize three distinct car-sharing systems which operate in Vancouver (Canada) and nearby regions, gathering data for more than one year. The study uncovers patterns of users' habits and demands for these services. We highlight the common characteristics and the main differences among car-sharing systems. Finally, we believe our study and data is useful for generating realistic synthetic workloads.

This chapter refers mostly the paper *Characterizing client usage patterns and service demand for car-sharing systems* Alencar et al., 2019, published on the Journal, available online since October 11, 2019. My contribution is mainly focused in sections [mike: mettere qui le sezioni] [mike: rivedere le cose I we e quali sezioni io ho effettivamente dato contributo]

4.2 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 the urban mobility, specifically the traffic-related mobility with motorized vehicles, is currently acquiring more importance, in order to improve the people quality of life studyng for example road mesh planning and communication resources allocation Herrera et al., 2010; Ma et al., 2013.

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, by 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 Ciociola et al., 2017. Therefore, it is important to collect and study the available open data and generate models that can help to understand the urban mobility and the social interactions of people in the urban environment.

Many alternative transport modes contribute to urban mobility. Among them, the car-sharing paradigm is quickly growing **becker2017comparing**; Boldrini et al., 2016; Ciociola et al., 2017. 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 Susan A. Shaheen, 2016. 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 and;
- 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 Boldrini et al., 2016

.

This chapter proposes a comparison between free floating and different station based car sharing paradigms. More in details we characterized those services in order to outstanding different users' habits. We take as 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

[1] service dismissed on February 29, 2020

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 accurate synthetic workload. As a consequence, these can contribute to the development of better capacity planning models to car-sharing systems and also to a better plan of public transport systems.

The remainder of this chapter is structured as follows: Section 4.3 describes related work; Section 4.4 describes details of the three car-sharing paradigms; Section 4.5 discusses the data collection and analysis methodology for all services; Section 4.6 presents the results of the characterization for each model and the comparison of them, whereas Section 4.7 concludes the paper.

4.3 Related Work

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 Becker et al., 2017; Boldrini et al., 2016; Francesco Ciari, Bock, et al., 2014; Martin and S. Shaheen, 2011. For example, one-way car-sharing systems are mostly used in dense urban areas with good public transportation system Stillwater et al., 2009. Young people with a higher education level are more attracted to use this service Burkhardt and Millard-Ball, 2006. Moreover, several works also confirm positive impacts on the actual transport system, such as the reduction on traffic and emission of pollutants Cervero and Tsai, 2004; Martin and S. Shaheen, 2011, the increase of free parking spots and in the use of public transport Susan A Shaheen et al., 2010. These prior works also reveal that one-way car-sharing services are used for long journeys and shopping Francesco Ciari, Bock, et al., 2014. In most cases, at least two passengers use the vehicle Becker et al., 2017. 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 Boldrini et al., 2016.

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 Francesco Ciari, Bock, et al., 2014, Becker et al., 2017 M. Cocca et al., 2019. Typically, free-floating vehicles carry a single user Becker et al., 2017 and this user presents fast driving habits Ciociola et al., 2017. 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 Ciociola et al., 2017. Despite the flexibility of the free-floating and one-way model, previous works have not observed a clear difference in users preferences between them Francesco Ciari, Bock, et al., 2014.

On the other hand, some works have identified that these services attract different users classes, exposing the fact that free-floating models and station-based models must be treated separately Becker et al., 2017.

To the best of our knowledge, only our prior works characterize the two-way car-sharing service model Rooke, Alencar, et al., 2019; Rooke, Aquiles, et al., 2018. More precisely, in Rooke, Aquiles, et al., 2018 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 Rooke, Alencar, et al., 2019, 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 Rooke, Alencar, et al., 2019. In M. Cocca et al., 2019; Michele Cocca, Giordano, Mellia, et al., 2019 we analyzed free-floating car-sharing data in different cities and propose models and optimization methods in order to efficiently use electric cars. We are not aware of studies that jointly study the three types of services in the same city, leveraging their common characteristics and highlighting its particularities as we are doing in the present work.

4.4 Car-sharing systems

The car-sharing systems might be implemented according 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 pick 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 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 to 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 to 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

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

car, parked into non-reserved spots in the streets. By the end of the use, users may leave vehicles at any location in a predefined area. Notably, free-floating model eliminates the limitations that station-based models hold, making the experience more flexible and closer to private-owned vehicles Francesco Ciari, Bock, et al., 2014.

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.

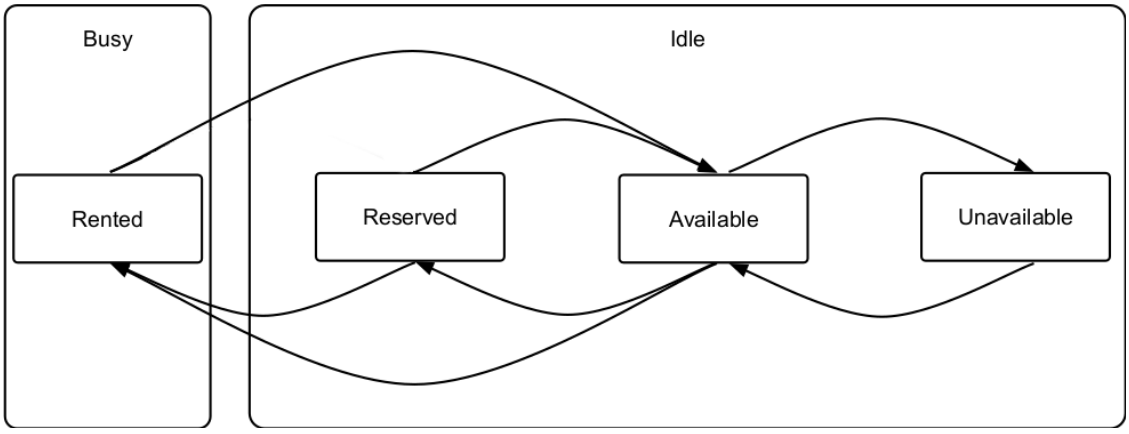


Figure 4.1: Possible states of a vehicle in a car-sharing system.

4.5 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 fleets 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 an 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 vehicles 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 the following trace repository: http://netlab.ice.ufjf.br/index.php/carsharingdata/

4.5.1 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

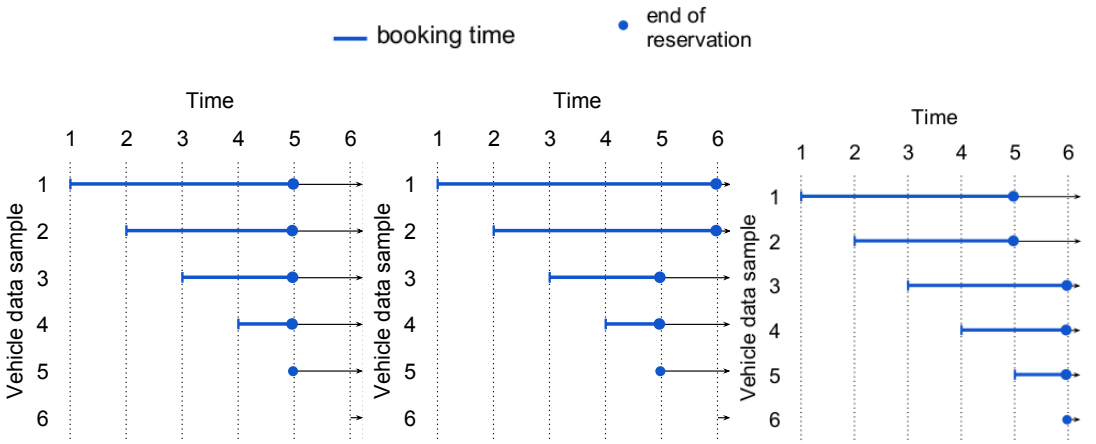
[3]Modo API, http://modo.coop/api/

[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

$t = 5$. 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 $t = 6$.



(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.

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 millions of 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 its neighbor cities. This data allows us to analyze more than 98 000 travels.[6]

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

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

Table 4.1: Summary of the Modo dataset.

4.5.2 Evo crawling methodology and data summary

Evo does not offer a public API to researchers. For this reason, we collect data which 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 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 behaviour and, as a consequence, differ 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 cancellation in Evo by filtering short travels (i.e., < 30 minutes) where the start and end points 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 its neighbor 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.

4.5.3 Car2Go crawling methodology and data summary

Car2Go offers APIs providing information about available cars at the moment of the request. Each API request returns, among other information, the car unique ID, its position and other fields which specifically describe the car status. Therefore the API response is semantically equivalent to the Evo's one. In this way, we applied the same methodology to gather and store the Car2go data too.

There are two main events, which changes the car status, clearly observable from the data. Considering the current time instant t_i:

- if in t_i the car is present in the API response and at time t_{i+1} it is not, that car passes from available to rented.
- 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. Indeed, for privacy constraints, the position of the car during a booking is not available.

Notice that from a single rented status is impossible to estimate the travelled distance: by computing the Euclidean or Haversine distance we obtain only a lower bound of the real travel distance which is practically too optimistic to be used as a primary travel estimation. To improve this estimation we attach to each entry the distance provided by the Google Maps API. As in Evo's methodology, we infer the number of cancellations by filtering short travels where the start and end points are very close. Table 4.3 summarizes Car2go dataset. We have more than one million travels in our 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

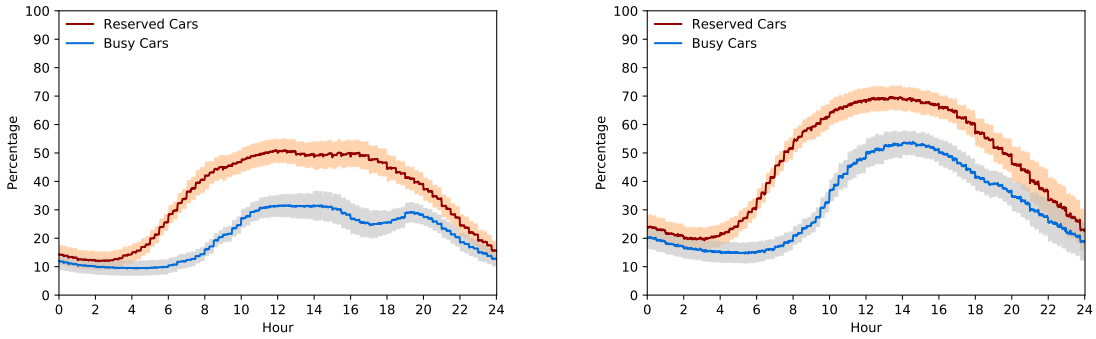
Table 4.3: Summary of the Car2Go data collection.

4.6 Car-sharing services characterization

In this section, we first present temporal characterization of the three services (Section 4.6.1). Then, we describe the services spatial-temporal characteristics (Section 4.6.2). Finally, we present users' behavior (Section 4.6.3).

4.6.1 Temporal characteristics

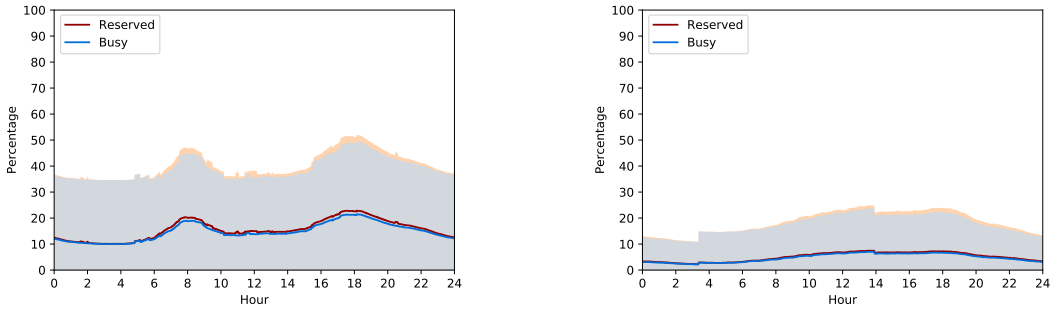
Figures 4.3, 4.4 and 4.5 show the service daily demand pattern. In all plots, the blue and red solid lines refer to a minute-by-minute mean value over the studied period for



(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



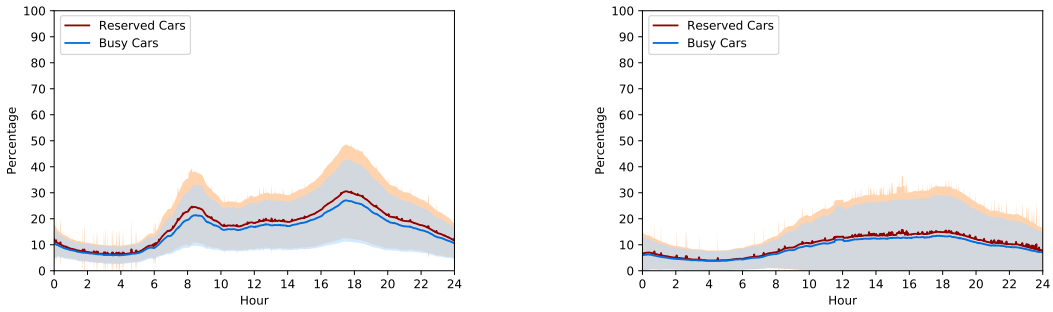
(a) One-way Evo Weekdays

(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

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 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



(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

weekly/daily/hourly subscription. In this sense, they tend to reserve cars at the same hour, for regular periods, which explains Modo 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%.

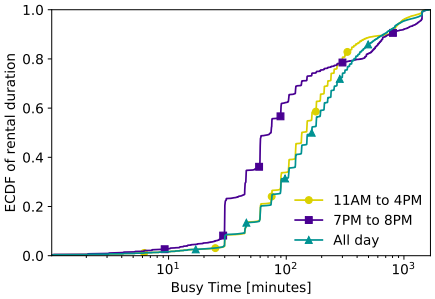
Figure 4.6 presents the Empirical Cumulative Distribution Function (ECDF) of vehicles busy time, i.e., the rental duration, during load peaks of the day. In this case, we evaluate the load periods from 7 AM to 10 AM and from 4 PM to 8 PM for free-floating and one-way, from 11 AM to 4 PM and 7 PM to 8 PM for two-way, and also all-day data for the three services.

As for the demand, Evo and Car2Go present similar behaviour, which is different from Modo. For Modo it is possible to observe at least 80% of vehicles rentals presents 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.

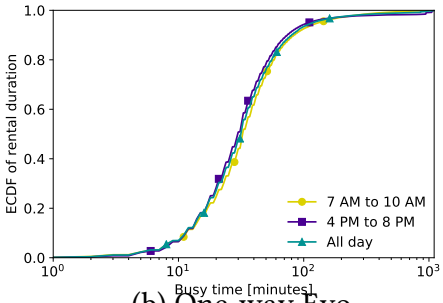
4.6.2 Spatial-temporal characteristics

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

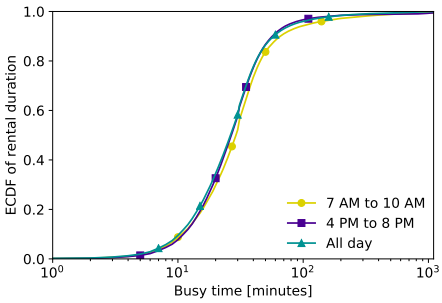
[7]Due to space constraints, we only show one-hour period every four hour.



(a) Two-way Modo

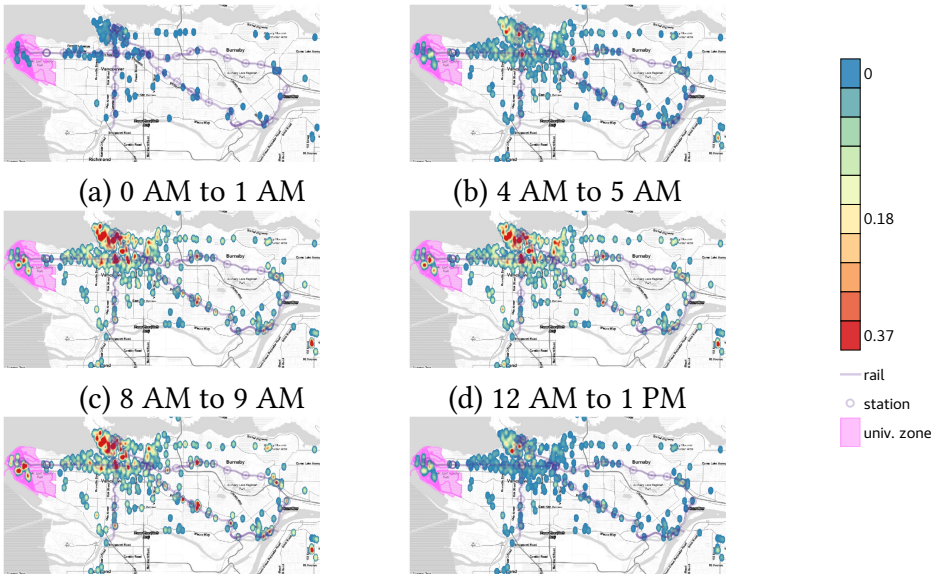


(b) One-way Evo



(c) Free floating Car2Go

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



(a) 0 AM to 1 AM

(b) 4 AM to 5 AM

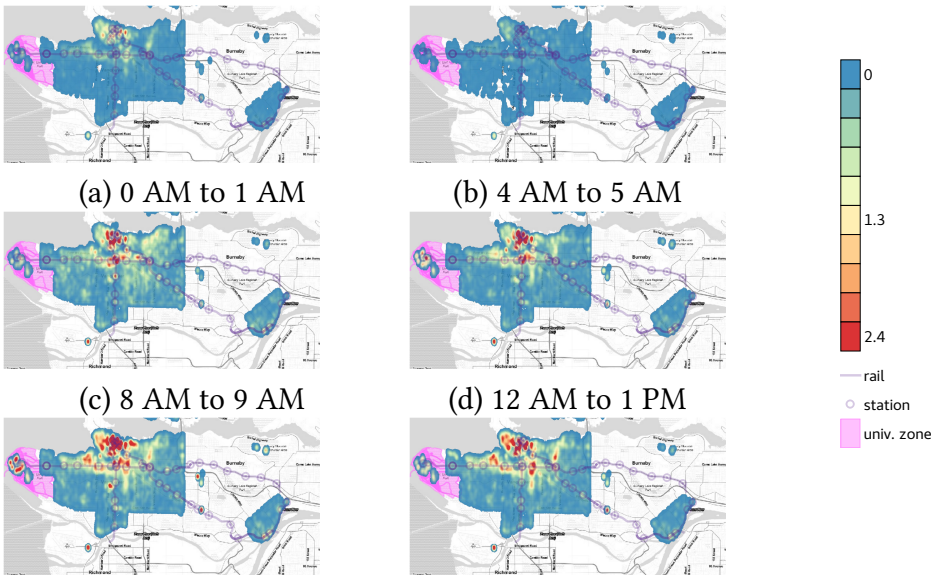
(c) 8 AM to 9 AM

(d) 12 AM to 1 PM

(e) 4 PM to 5 PM

(f) 8 PM to 9 PM

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

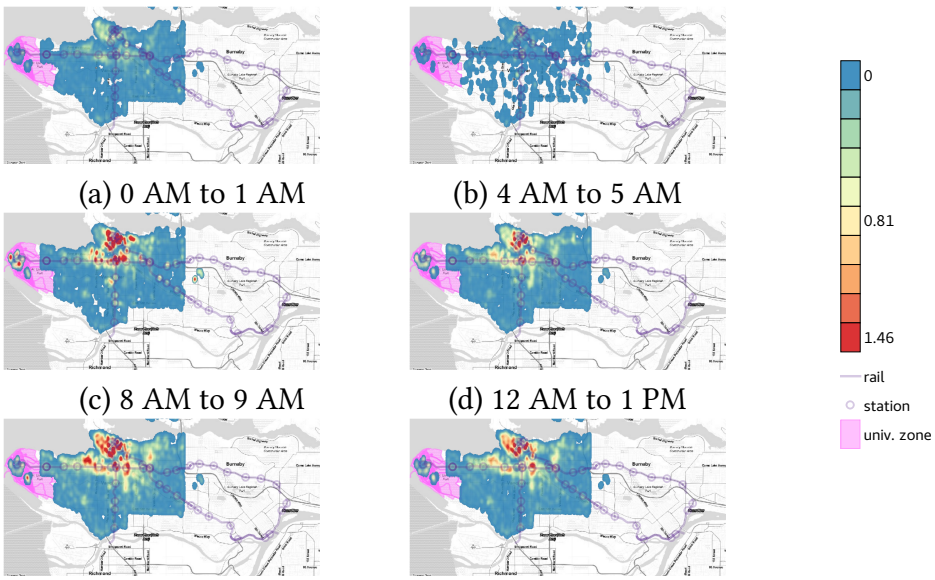


(a) 0 AM to 1 AM (b) 4 AM to 5 AM

(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 4.8: 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.

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. Rangin radius from 100 m to 1000 m, leads to similar results.

First, all three services present a large demand in the downtown area and the university zone. Note that the demand in 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 to 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 in downtown, it is present a widespread demand all over the city.

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 constraint, 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 to spread services along all city area is noticeable. Moreover, we also note that some regions serve as hubs. This is more notable for Evo 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 at 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.

4.6.3 User behavior characteristics

Vehicles 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

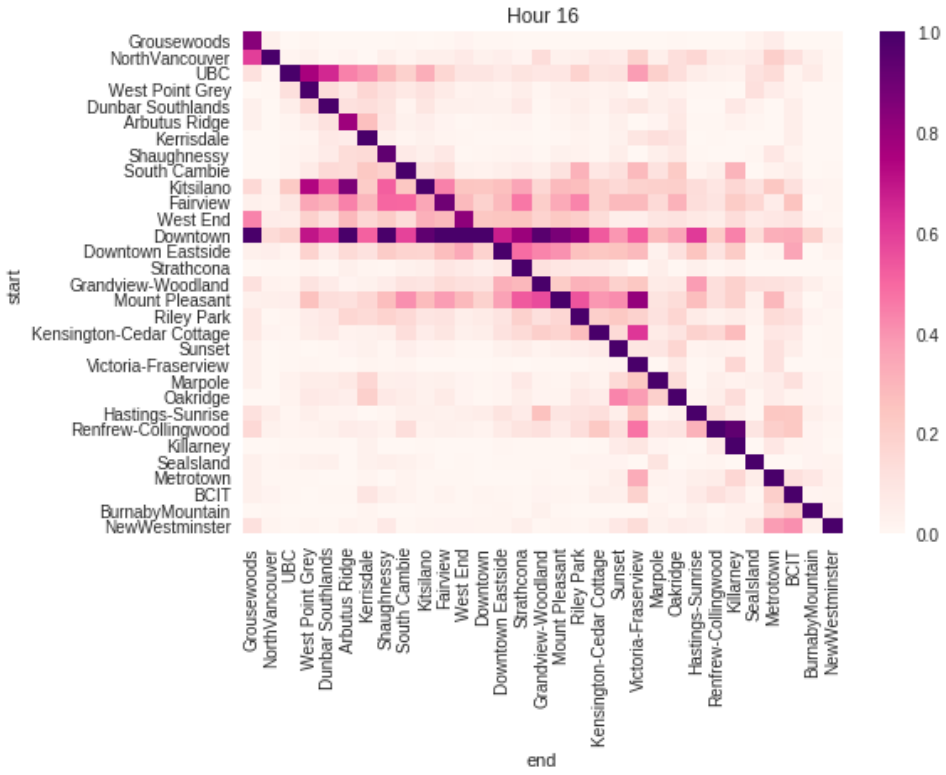


Figure 4.10: Origin-destination matrix for one-way service Evo (from 4 PM to 5 PM).

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

[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: $F(x, k, s, \beta, \delta) = \frac{((x-\beta)/\delta)^k}{(1+((x-\beta)/\delta)^s)^{(k*\frac{1}{s})}}$

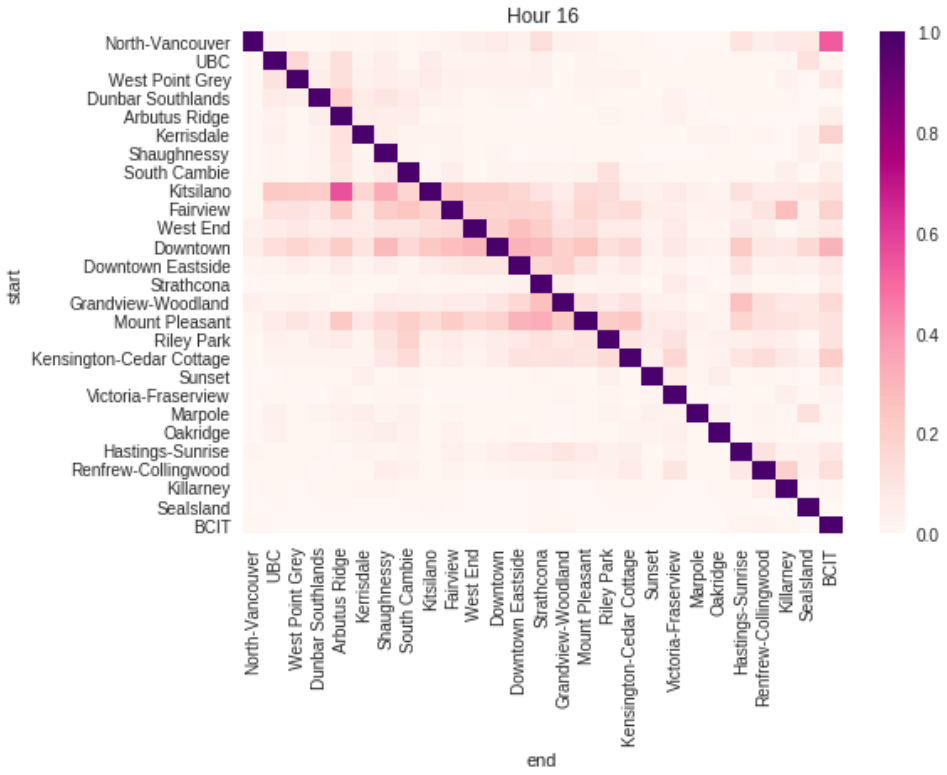


Figure 4.11: Origin-destination matrix for free-floating service Car2Go (from 4 PM to 5 PM).

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 to our previous observations. Indeed, the demand for car-sharing varies over the city during a 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

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

remains idle for the same period.

In sum, our 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.

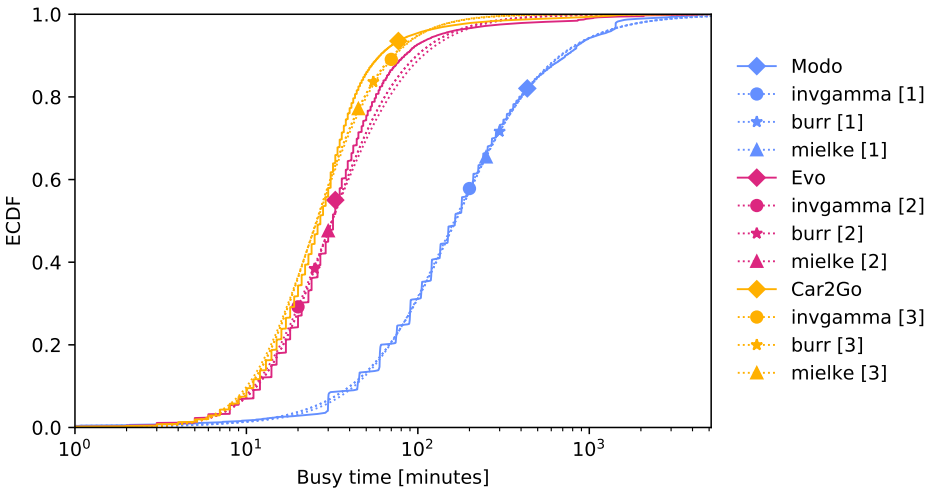


Figure 4.12: Cumulative distribution function of vehicle busy time.

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.

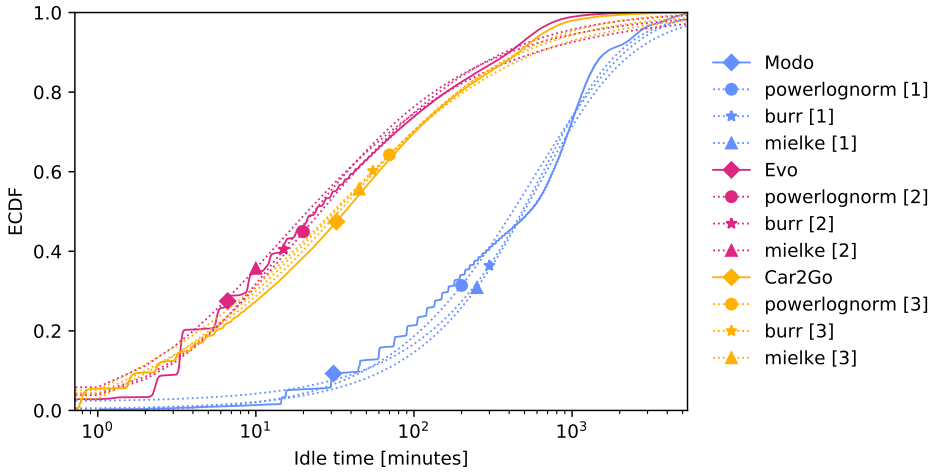


Figure 4.13: Cumulative distribution function of vehicle idle time

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.

4.7 Conclusions

In this article, we characterized three distinct car-sharing systems which operate in Vancouver (Canada) and nearby regions. Our study, using data of more than one year of real trips, 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 points 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 in weekends, peaks are distributed in the afternoon. The two-way service we analyze 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 interests such as public universities and commercial centers is present.

Chapter 5

Simulator

- introduzione, dove spiego la necessita di un simulatore
- struttura del simulatore + algoritmo
- chartging station palcement + grafiici (avg time, num parking)
- charging policies

5.1 Abstract

in this chapter I describe mainly the simulator ...

5.2 Introduction

5.3 Electric car sharing simulator

The goal is to study different design choices for electric car sharing systems. 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 by a subset of rentals collected in 2. In this way, by implementing an electric car consumption, I am able to model an electric FFCS provider that exactly replicate customers' temporal and spatial demand.

5.3.1 Simulation model

The simulator replicates the behaviour of a fleet of electric cars, which are moving in the city. Each car is characterized by its location, and the current status of 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, each trip $i \in \mathcal{I}$ is characterized by its start and end time, $t_s(i)$ and $t_e(i)$, and origin and destination coordinates, $o(i)$ and $d(i)$. For simplicity, I divide the city area into squared zones, of side 500 m as before. Then, I associate with each position one and only one zone $O(i) = zone(o(i))$ and $D(i) = zone(d(i))$. We assume a charging station cs, composed of k poles, can be placed at the center of a given zone $z \in \mathcal{Z}$, so either $cs(z) = 1$ if the station is present, or $cs(z) = 0$ otherwise. $N = \sum_{z \in \mathcal{Z}} cs(z)$ is the total number of zones equipped with charging stations, with $K = N \cdot k$ the total number of poles.

Additionally, it is present a set $\mathcal{A}$ of cars, with its cardinality $|\mathcal{A}|$ obtained by the trace. Each car $a \in \mathcal{A}$ at time t is characterized by its position $p(a,t)$, its zone $P(a,t) = zone(p(a,t))$, and the residual battery capacity $c(a,t) \in [0, C]$, 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 $\bar{a} \in \mathcal{A}$ such that:

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

Basically, the simulator mimics the normal behaviour of FFCS customers that use their smartphone to rent the closest car from their position and are worried about vehicle range Jung et al., 2015. Notice that this behaviour is independent from 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 output, the simulator produces several statistics about system usage and user-related discomfort metrics related to the electric vehicle plugging procedures.

5.3.2 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 $a \in \mathcal{A}$ such that

$$P(a,t) = O(i) \; \wedge \; c(a,t) \geq c(a',t) \; \forall a' \mid P(a',t) = O(i),$$

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

[1] We choose this policy because people are worried about vehicle range Jung et al., 2015.

If any car is available, the simulator selects the closest zone to $O(i)$ containing an available car, mimicking the normal behaviour of FFCS customers that use their smartphone to rent the closest car from their position. If any vehicle is present in the 8 eight neighbouring 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 re-routing due to vehicle plugging procedures. The policies to decide when and how plug the car are described in section [mike: define sezione]. 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:

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

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.

5.4 Meta-Heuristic Charging Stations Placement

In this sections I explain the ratio behind the charging station placement. Indeed, in the previous section I described the structure and simulation policies, while now I define an important environmental variable that will be deeply studied in chapters [mike: references]. I remind that from the rental booking trace, I extrapolate the service area and I divide it into square zones of 500 m of side length. Then, using the number of parking or the average parking time for each zone, the simulator choose which zones equip with a charging station.

5.4.1 Problem formalization

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

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

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 an analytical general solution. Moreover the number of candidate solutions increases as the binomial coefficient $\binom{|\mathscr{Z}|}{N}$, making ineffective to numerically compute all possibilities. Instead, I will provide a class of greedy algorithms and analyse 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,

$$\max \sum_{z \in \mathcal{Z}} cs(z) l_z$$

subject to:

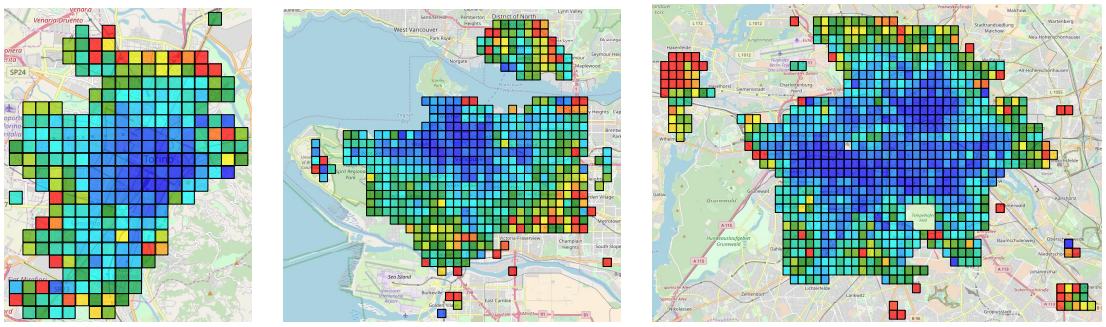
$$\sum_{z \in \mathcal{Z}} cs(z) = N$$

$$cs(z) \in \{0,1\}, \forall z \in \mathcal{Z}$$

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 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;

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



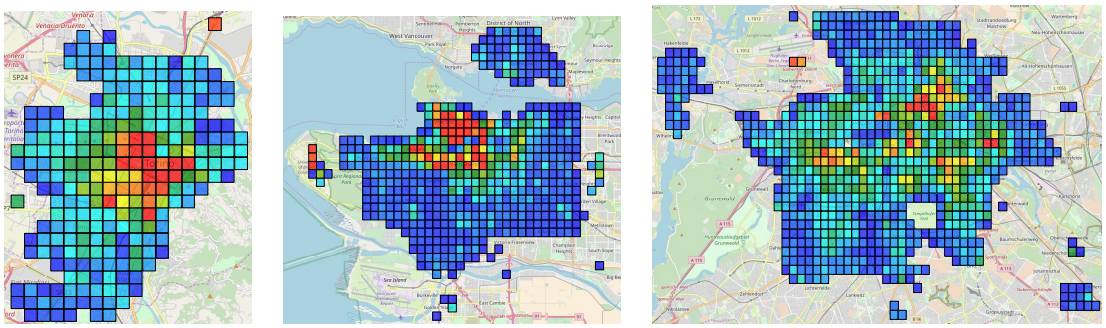
(a) Turin (b) Vancouver (c) Berlin

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, in particular, 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 se how, in all figures, the heuristic *Average parking time* is mainly spread in city peripheries. It means that the cars spend a lot of times parked far from city centre. 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 high-density residential area located in periphery. This, joint with the low business-days night demand, leads to users to leave cars parked in that areas all night long.



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

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

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 parkings means an higher zone attractiveness, it is possible to notice how the zones with the highest number of parking concentration zones 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 actractive areas. This is mainly due to the biggest operative area and, probably, to the differentiation of business areas.

This brief catheterization shows how different cities can have different spatial characterization and thus different charging station placements. However those characterization will be deepened in chapters [mike: captoli].

5.5 Users' Plugging Policy

One of the most challenging points of electric FFCS is to deal with the discomfort derived by plug in operations. This operation is more time consuming with compared to the normal filling up procedure of combustion engine cars. Therefore, the providers have to deal with users' selfishness and trying to stimulate their willingness. In this Section, we define different policies that the electric FFCS may enforce, and different probabilistic behaviors of customers.

5.5.1 Car charging policies

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 modelled 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, i.e., $(c(a, t_s(i)) - Energy(p(a, t_s(i)), d(i))) \cdot 100/C \leq \pi$. 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 battery is close to exhaustion. It used as benchmark, in order to understand until when the users might rent 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 willingness 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 by justified like provider incentive bonus like car2go free minutes after a car filling up.

5.6 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 of users discomfort.

5.6.1 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 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 $c(a, t)$, and fraction of time a battery stays under charge.

5.6.2 Simulation scenario

I used this simulator to study the impact on 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 travelled 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 then 5 seconds to complete a single simulation for a given city and parameter set. Due to the 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.

5.7 Conclusion

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 trace of rental events, describing the system allocated users' demand.

More in details, the simulator allocates a set of cars, characterized by battery capacity and power consumption per kilometres. 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 to place the

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

charging station according three heuristics: random, preferring zones having a grater parking time and zones having the higher number of parkings. Finally it takes in 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 vehcile

Chapter 6

Metaherustic charging station placement in 4 cities

6.1 Abstract

Car sharing is a popular means of transport in smart cities, with the free floating paradigm which lets the customers autonomously pick and drop available cars. This freedom impose several challenges and trade-off in the design of an electrical version of this type of car sharing.

In this paper we study the design of an electric free floating car sharing system for four cities by leveraging actual rental traces. We analyze via accurate simulations the impact of i) the charging station placement, and ii) return policies. Our traces composed by millions of rentals in different cities world wide located, captures the non-stationary and highly dynamic nature of usage patterns of customers.

Considering charging station placement, we demonstrate how it is better to place charging stations within popular parking area (e.g., downtown), even if parking duration is short. We show how by smart placing and handful number of charging stations (below 10% of city zones) the system can self-sustain itself. These results are obtained also thanks to customers' willingness to collaborate by voluntarily returning the car to a nearby charging station.

Our results help car sharing provider comparing possible alternative design solution, i.e., the adoption of simple relocation policies that would move cars that need a charge only, a promising solution to limit discomfort for customers due to rerouting. The same could be achieved by considering giving incentives to customers. We offer all data and tools used in this paper to foster further studies in these directions.

6.2 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 mean to improve the current situation. Along with the usage of public transport, the sharing mobility such as bike sharing, car pooling and car sharing, can help to address this problem. In this work, we 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. We 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 geo-fenced area. Electric car sharing systems need an infrastructure of recharging stations, whose design requires ingenuity Chen et al., 2013; Jiao et al., 2016; Phonrattanasak and Leeprechanon, 2014. 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 asking the customer to return the car to a charging station.

Data is fundamental to answer these design questions. In this work, we base our study on the availability of millions of actual rentals we collected from currently in use FFCS systems. In the past we focused on the study of the city of Turin (Italy) Michele Cocca, Giordano, Vassio, et al., 2018. Here, we generalize our approach to other cities in different countries, while extending and deepening our findings. In this paper, we leverage real FFCS data related to the city of Vancouver (Canada), Berlin (Germany), Milan, and Turin (Italy). Our data naturally factors the non-stationarity of FFCS systems including millions of actual rentals. Armed with this, we 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 Jorg Firnkorn and Muller, 2015; Weikl and Bogenberger, 2015 and for a smart placement of charging stations, we are among the first to take a data driven approach for the design of electric car FFCS systems Chen et al., 2013; Hess et al., 2012; Jiao et al., 2016; Phonrattanasak and Leeprechanon, 2014.

First, we characterize how customers actually use FFCS transport means in different cities and countries. Results show a similar usage with an high utilization during 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, we observe peripheral zones where cars are left parked for long time, and busy areas where instead the parking duration is much short and dynamic.

Armed with these facts, we compare charging station placement policies that exploit the knowledge of typical parking zones and duration. We 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 long time performs badly. Instead, placing charging station 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, we study different return policies, 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 a pure opportunistic free floating solutions. Equipping just 8% of charging zones with 4 poles of 2 kW would guarantee a electric car FFCS equivalent to the one currently in use. This with minimal impact on the 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, we compare system design alternatives to check whether is better to place a lot of charging poles in very few areas, or rather to spread a lot charging stations with few poles in many areas. Results demonstrate that both extreme solutions perform badly, with 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.

We believe results presented in this paper, guided by actual usage pattern for FFCS customers, are very important for regulators and policy makers, as well as for researchers working in this area. In an effort to allow reproducibility and extend our results, we make both dataset and simulator publicly available.[1]

After quickly discussing related work in Section 7.3, we present and characterize data in Section 7.4, and the simulation model and tool in Section 7.5. Section 7.7 discusses the impact of charging stations placement policies, while Section 6.7 compares return policies. Section 6.8 presents the impact of concentrating or spreading charging stations in the city. Finally Section 7.11 concludes the paper.

6.3 Related work

The diffusion of the free floating approach to car sharing leaded to an increasing attention by many researchers, with analyses of these systems and their extension to electrical vehicles. The studies performed in 2011 by Finkorn and Müller Jörg Firnkorn and Müller, 2011, 2012 are the first attempts to analyze benefits of FFCS for the population. Their results on customers' behavior, like traveled distances, are similar to ours, and our first work **UMAP**. Later works **CSandSocial**; Habibi et al., 2016; Schmöller et al., 2015 also collected data and analyzed the mobility patterns of customers and differences among cities.

The introduction of electrical vehicles for private and public transportation, brought the problem of placing the electric charging stations. Authors in Hess et al., 2012 show the benefits of placing charging stations with different power according to the customer

[1] https://github.com/michelelt/sim3.0

parking duration. Few data driven studies address the charging station placement, either by respectively minimizing cost of installation, power loss and maintenance Jiao et al., 2016; Phonrattanasak and Leeprechanon, 2014, or by minimizing the customers' walked distances necessary to reach a charging poleChen et al., 2013.

After a survey among FFCS customers in Ulm (Germany), authors of Jorg Firnkorn and Muller, 2015 investigate the positive influence and feasibility of an electric FFCS systems. Lastly, authors of Weikl and Bogenberger, 2015 study the relocation of electric cars in FFCS, since few charging stations may be blocked by completely charged vehicles.

Previously, in Michele Cocca, Giordano, Vassio, et al., 2018, we performed several analysis about how to design an electrical FFCS in the city of Turin. In Michele Cocca, Giordano, Mellia, et al., 2018 we introduced a first optimization of electric charging stations placement. In this work, we extend both works, by considering 4 cities as case study, and studying a new set of return policies to observe the impact of the willingness of customers to contribute to the system sustainability. We further extend our work by discussing the benefits of using charging hubs.

For our knowledge, in this work we are 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 word wide located.

6.4 Data collection and characterization

Here we describe the methodology we followed to harvest data from already operative FFCS systems in the city of Turin (Italy), Milan (Italy), Berlin (Germany), and Vancouver (Canada). We first describe the data collection mechanisms. Then we characterize system usage, focusing on those metrics that are instrumental for the design of FFCS systems based on electric vehicles and highlighting the non-stationary patterns.

6.4.1 Data collection and filtering

The modern FFCS provider like Car2go, DriveNow or GoGet use information systems to expose the position of the available cars through web services. Customers access these to check which cars are available for a rental by using a smartphone app. For instance, Car2go offers public Application Programming Interface (API) [2] to access the information about the system status. We use these APIs to collect data about actual rentals occurred over time following a streaming approach. First, we get the operative area for each city, i.e., the zone where cars can be rented and returned. The operative

[2]car2goAPI, `https://www.car2go.com/api/tou.htm` service subject to approval by car2Go. Approval granted in Sept. 2016 and discontinued in Jan. 2018.

areas remain constant for several months. Next, we collect a snapshot every minute reporting the positions of all cars available for a rental, for each city. From each snapshot we save the car plates (used as car identifier) and its geographic coordinates.

We developed UMAP **UMAP**, a software able to process these snapshots and rebuild, for each car, the history in terms of *bookings* and *parkings*.[3] A booking is an event describing a possible car rental i, characterized by the starting and final position $o(i)$ and $d(i)$, and starting and ending time $t_s(i)$ and $t_e(i)$. A parking describes a period of time when the car was parked and available for rentals; it is characterized by the location and the starting and final time. In a nutshell, we process the snapshots of the available cars and derive booking periods by observing which cars are available at each snapshot. Given a car, when it "disappears" from a snapshot, it means that someone has booked it – so that the car is not available for other customers. A new booking is started. When the same car "reappears" back later on, it means someone has returned it. The booking is then ended.

Not all bookings correspond to actual rentals: (i) a customer can book a car, and cancel the booking later on; (ii) the data collection may suffer from outages, so that some snapshots may miss some available cars; (iii) cars may go in maintenance, so that they disappear and never come back (or return after a long time). We developed data cleaning and filtering rules to extract actual *rentals* from bookings. A booking is considered a valid rental if: (i) it lasts at least 3 minutes; (ii) the ending position is at least 700 m far from the staring position, with both positions inside the city operative area;[4]; (iii) its duration is smaller than 1 h. These thresholds have been selected by domain knowledge of the data – see **UMAP** for more details. Bookings that do not correspond to rentals are then merged with parkings events (since the car did not move).

We started to collected data in December 2016 in all the 22 cities in which Car2go is present, and stopped our collection in February 2018. Overall, we obtained more than 28 millions rentals and parkings that we leverage for our studies.

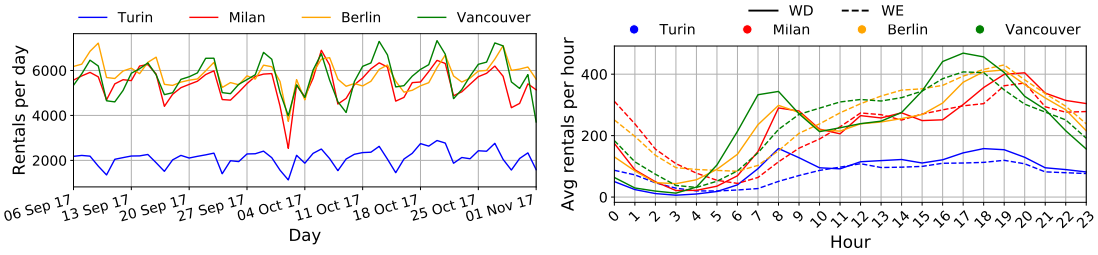
6.4.2 Temporal characterization

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

To give the intuition of the system, we first provide a characterization of actual usage patterns by current FFCS customers in each city. We focus first on the temporal characteristics. Fig. 6.1 reports the rental trend. More in details, Fig. 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.

[3]UMAP can be downloaded from https://github.com/MobilityPolito/UMAP

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



(a) Number of rentals per day.

(b) Average number of rentals per hour.

Figure 6.1: Rental trends in different cities.

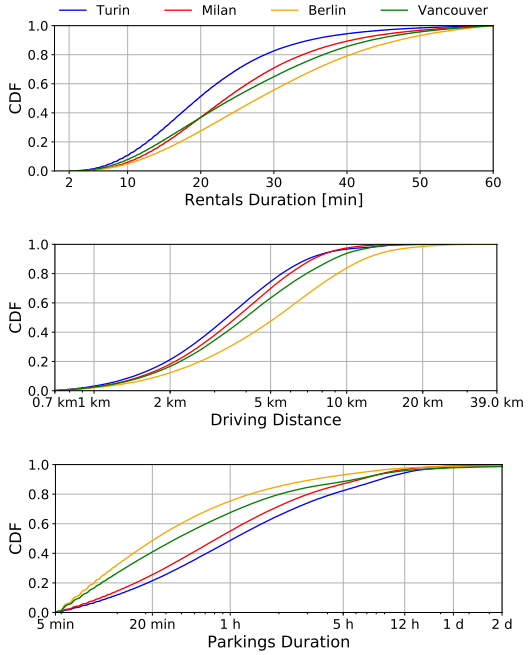
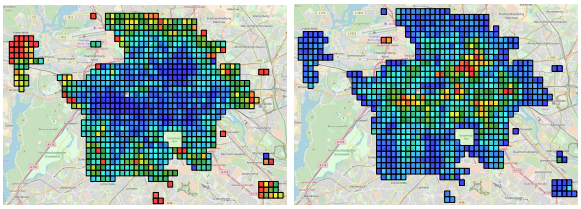
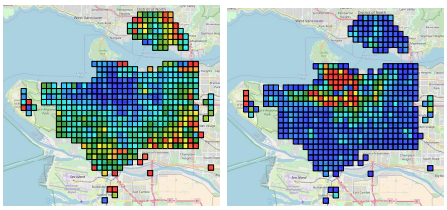


Figure 6.2: Car Sharing usage habit characterization.

This intense usage, justify the difference in fleet size between the cities, with these three having at least twice as much cars with respect to Turin (see Tab. 6.1). These first results highlights the importance of extending our previous work to other cities than Turin. A second interesting aspect is the presence of a weekly pattern: in correspondence of the weekends the number of rentals drops of 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



(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.

transport strike.[5]

To deeply analyze customers' habits, we detail the average number of rentals per hour in Fig. 6.1b, separately per working-days (WD, solid line) and per weekends (WE, dashed line). Each curve reports the number of rentals for each our, 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., 8am 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 of pubs and nightlife. At last, observe again the different patterns 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 Italian cities. The charging station placement design must thus weight this different needs and non-stationary patterns.

Given our goals of deriving guidelines for charging station placement policies, we 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 we have no information on car position during a rental, we 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). We use Google Map service to compute a correcting factor to be applied on the euclidean distance **UMAP**.

Fig. 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

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

a clear impact, with Turin that has the shortest trips, and Berlin the longest. 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 Fig. 6.2, we 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, bottom plot of Fig. 6.2 details the duration of the parking periods. Interestingly, 50% of parkings lasts less than 22 minutes in Berlin, testifying a very high system utilization. In Turin, the median grows to 42 minutes, still showing that most cars are parked for short time. Yet, the long tail of the CDF (note the log scale on x-axis) suggests that there is a sizable 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 commuting time and for short trips.

6.4.3 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 long time – so to maximize the charging time. Knowing the zones within the city where cars are left parked is fundamental. For this, we divide the operative area of each city into squared zones, obtained with a grid of 500 m of side. For each zone, we compute the total number of parkings and the average parking time.

Fig. 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 - Fig. 6.3a. Left plot shows the average parking time. Results clearly show that cars stay parked for very short time in busy downtown areas, while cars stay

parked for longer time in the periphery (corresponding to the head and tail of the CDF in Fig. 6.2, respectively).

Conversely, right plot depicts the total number of parkings recorded in each zone. It clearly shows that areas where the majority of rentals/parkings occurs 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 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 we 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.

6.5 Electric car sharing simulator

Our goal is to study different design choices for electric car sharing systems. For this, we developed a flexible event-based simulator that allows us to compare different algorithms and tune their parameters while collecting metrics of interest. Simulations are based on the actual traces collected from operative FFCS providers in each city. This allows us to factor all spatial and temporal characteristics of actual FFCS customers habits.

6.5.1 Simulation model

We simulate a fleet of electric cars, which move in the city. Each car is characterized by its location, and the current status of 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, each trip $i \in \mathcal{I}$ is characterized by its start and end time, $t_s(i)$ and $t_e(i)$, and origin and destination coordinates, $o(i)$ and $d(i)$. For simplicity, we divide the city area into squared zones, of side 500 m as before. We associate with each position one and only one zone $O(i) = zone(o(i))$ and $D(i) = zone(d(i))$. We assume a charging station cs, composed of k poles, can be placed at the center of a given zone $z \in \mathcal{Z}$, so either $cs(z) = 1$ if the station is present, or $cs(z) = 0$ otherwise. $N = \sum_{z \in \mathcal{Z}} cs(z)$ is the total number of zones equipped with charging stations, with $K = N \cdot k$ the total number of poles.

We have a set $\mathcal{A}$ of cars, with its cardinality $|\mathcal{A}|$ obtained by the trace. Each car $a \in \mathcal{A}$ at time t is characterized by its position $p(a,t)$, its zone $P(a,t) = zone(p(a,t))$, and the residual battery capacity $c(a,t) \in [0, C]$, with C being the maximum nominal capacity.

The simulator processes each rental event i in temporal order. When a *rental-start* event i is processed at time $t = t_s(i)$, we choose 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:

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

Basically, we mimic the normal behavior of FFCS customers that use their smartphone to rent the closest car from their position and are worried about vehicle range Jung et al., 2015. Notice that this behavior is independent from whether the car is at a pole being charged or not.

A *rental-end* event is then scheduled using the trace final time $t_e(i)$ and desired destination location $d(i)$.

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

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

independently whether the car is at a pole being charged or not.[6]

If no car is present, we select 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. 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 $t_e(i)$, we return the car in a real position $p(a, t_e(i))$, chosen according to the behavior described in Section 6.5.2. The simulator updates the battery charge status by consuming an amount of energy proportional to the real trip distance:

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

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.

6.5.2 Car charging policies

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.

[6] We choose this policy because people are worried about vehicle range Jung et al., 2015.

In this Section, we define different policies that the electric FFCS may enforce, and different probabilistic behaviors of customers. We investigate 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, i.e., $(c(a, t_s(i)) - Energy(p(a, t_s(i)), d(i))) \cdot 100/C \leq \pi$. 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 battery is close to exhaustion. *Forced* mandates to connect cars to a charge station only when energy runs low, thus trying to protect from battery exhaustion. *Hybrid* balances the two policies, with w that measures the level of customers willingness to collaborate. $w = 0$ 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.

6.5.3 Charging stations placement

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

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

Since we do not make any assumption on the set of trips $\mathscr{I}$, we cannot know a-priori if a solution exists and provide an analytical general solution. Moreover the number of candidate solutions increases as the binomial coefficient $\binom{|\mathscr{Z}|}{N}$, making ineffective to numerically compute all possibilities. Instead, we 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,

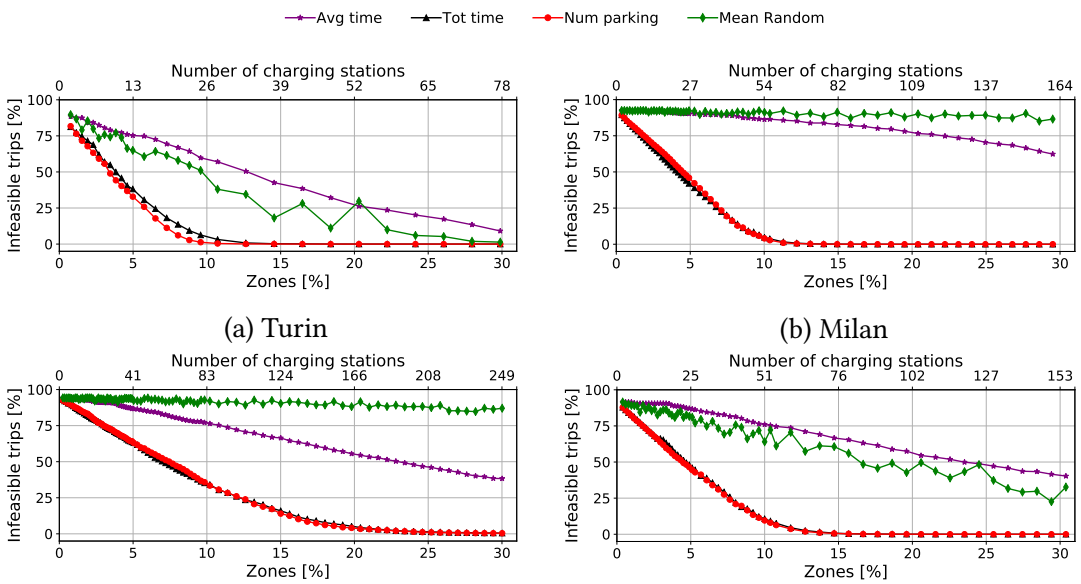
$$\max \sum_{z \in \mathscr{Z}} cs(z) l_z$$

subject to:

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

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

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 different definition of the likelihood.



(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.

- *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.

As discussed in Section 6.4.3, the last three heuristics are driven by the intuition that placing charging stations in those zones where cars are parked for long time (average parking time) or frequently parked (total number of parkings) could improve system performance.

6.5.4 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 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 $c(a, t)$, and fraction of time a battery stays under charge.

6.5.5 Simulation scenario

We focus on key design parameters, i.e., the charging station placement and return policies. We then study the impact on the number of zones that are equipped with charging stations N, and the number of poles k of each charging station.

Table 6.1 summarizes the dataset main characteristics. We 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 $kWh/100\,km$). [7] 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.

Our simulator, written in Python, takes less then 5 seconds to complete a single simulation for a given city and parameter set. Due to the 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.

6.6 Impact of charging station placement

We consider the Free Floating return policy, and study the impact different charging station placement policies. Our aim is to check what would be the minimum number

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

of charging stations to install to sustain a FFCS system based on electric vehicles that is equivalent to the one currently in use.

Fig. 7.3 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. Bottom x-axis reports the percentage of equipped zones with respect to the total, while top x-axis reports the actual number, different for each city.

We observe notably different performance for different placement algorithm. 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 – Fig. 6.4b – and Berlin – Fig. 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 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 city but Berlin we can reach a negligible percentage of infeasible trips with just 15-18% of charging zones. Instead, in Berlin we still have some infeasible trips with 30% of charging zones. Recalling that with our battery we can travel 135 km, the presence of infeasible trip is explained by looking the rental distance presented in 6.2. Trips in Berlin can be as long as 39 km. Therefore, with only 4 long-trips which 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 favors 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 Fig. 6.3a and Fig. 6.3b). On the contrary, *Num parking* and *Tot time* favor city center areas, where cars are frequently parked for short time (see right heat maps in Fig. 6.3a and Fig. 6.3b).

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, we will use the total number of parking placement algorithm for the rest of the paper.

6.7 Impact of return policy

We now investigate the impact of the different return policies. In particular, we 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.

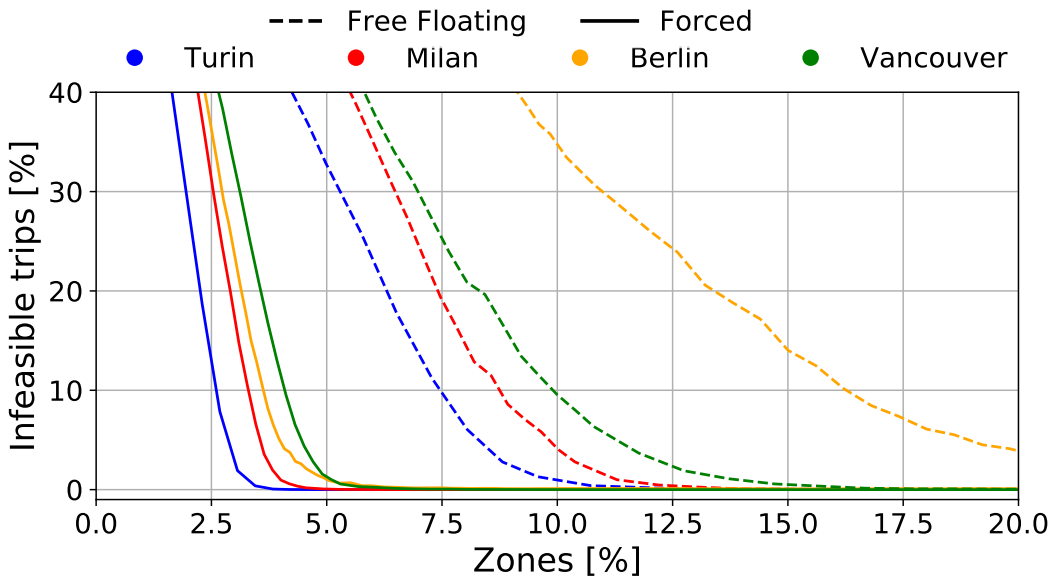
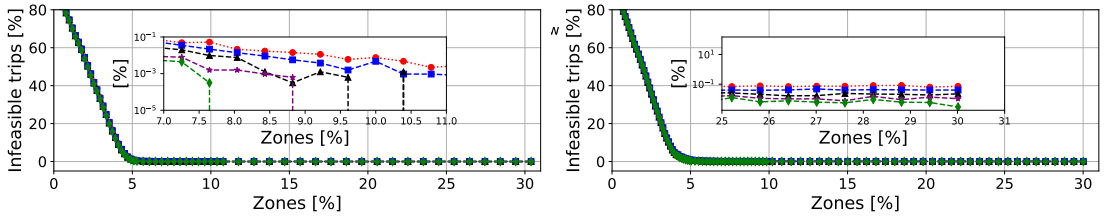


Figure 6.5: Infeasible trips when comparing the Free Floating vs Forced return policies. Forcing customer to charge when $c < \pi$ drastically improves performance.



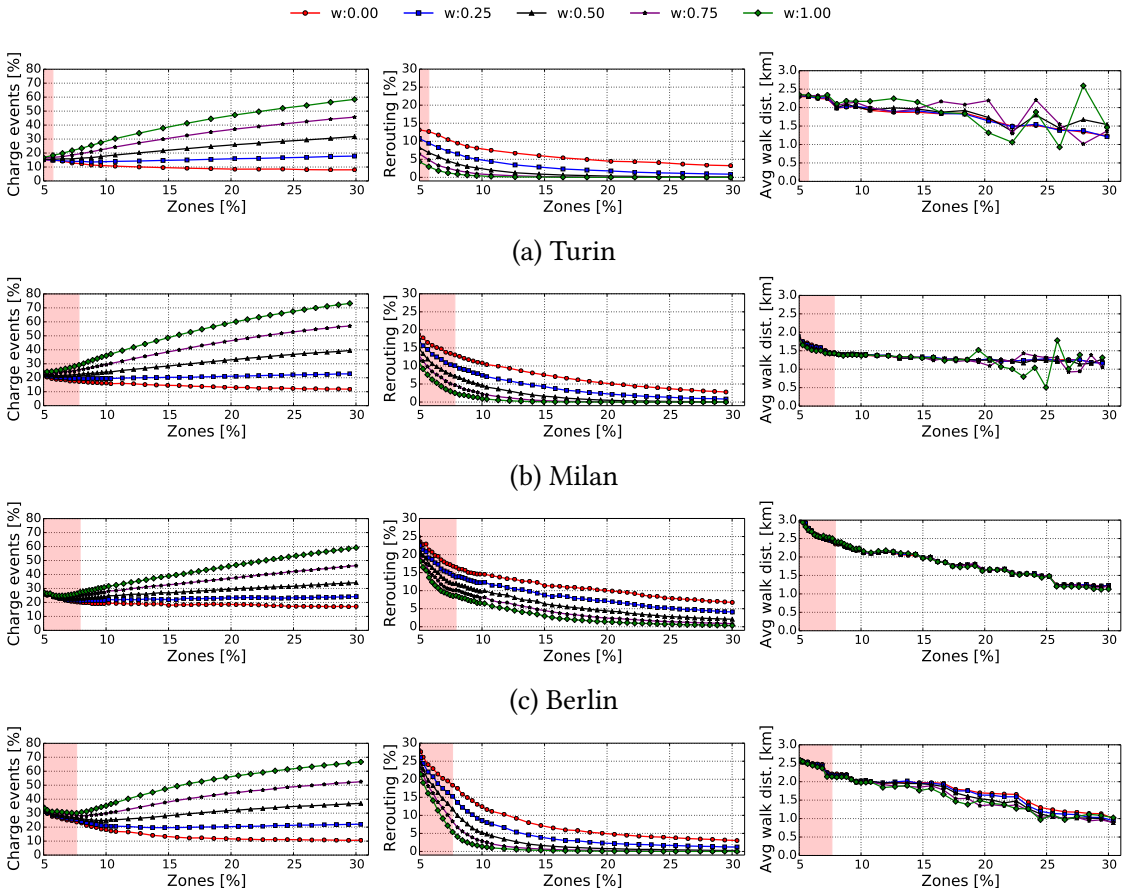
(a) Vancouver

(b) Berlin

Figure 6.6: Impact of customer willingness to cooperate w. Albeit the small impact, the higher w is, the better it is.

6.7.1 Impact on infeasible trips

Fig. 6.2 we already noticed that trips are typically limited. This is instrumental to choose a proper minimum charging threshold π. Given the maximum trip distance, we obtain the corresponding maximum energy being consumed. For instance, a maximum distance of 20 km correspond to about 15% of battery capacity for the considered car model. In the following, we take a conservative approach and set the minimum battery charge threshold $\pi = 25\%$. To make results comparable, we keep the same threshold also for Berlin, where the maximum traveled distance grows to 39 km, i.e., suggesting $\pi \geq 30\%$. Here our choice is not conservative.



(a) Turin

(b) Milan

(c) Berlin

(d) Vancouver

Figure 6.7: Charge events percentage (left), reroute events percentage (middle) and walk distance averaged over rerouted trips (right), per city. Increasing w benefits the customers' experience by reducing the rerouting events significantly, but increasing the charging events.

As before, we focus on the infeasible trips percentage with respect to N. We compare results for the Free Floating and the Forced policies. Fig. 6.5 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 which mandates customers to charge the car when battery level gets below a threshold drastically reduces the number of infeasible trips, even with a handful of charging stations. Indeed, in all cities we have a negligible percentage of infeasible trips ($< 0.1\%$) with more than 8% of zones equipped with charging stations.

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

system.

Fig. 6.6 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, we 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, we can still observe some infeasible trips 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 zone covered by charging stations, the systems becomes in all the cities almost self-sustained.

6.7.2 Impact on customer experience

As there is no strong evidence that the overall system would perform better with customers' altruistic behavior, here we 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, we show indexes that involve customers' experience.

Fig. 6.7 reports, for each city, and for different willingness the percentage of charge events (left plots), the percentage of reroute 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 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. 6.8a.

Interestingly, the charge percentage decreases for a selfish customer ($w = 0$), reaching about 5-8% for sufficiently large number of charging stations. This corresponds to the average number of charges per car that guarantees minimum battery charge of 25%. Indeed, given the average rental distance of 5 km (Table 6.1) and 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.

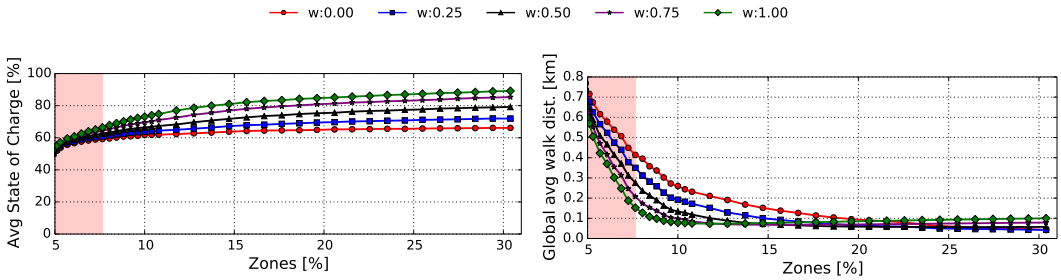
Move now to the percentage of rerouting events - middle plots in Fig. 6.7. Two important effects are visible. First, rerouting probability decreases as expected: the more the stations are, the more likely customers find a charging station at their desired final zone. With selfish behavior ($w = 0$), the fraction of rerouting events remains large even for large N. Second, the more collaborative are customers, the better it is for the entire system. With half of customers that return voluntarily the car to charging station if present in their final destination, for Milan, Turin and Vancouver the reroutings 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 maximum 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 Fig. 6.7. 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 rerouted.[8] 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, we would like to have $w = 1$ to reduce (far) reroutings and infeasible trips, and $w = 0$ to reduce charging events. In an attempt to take into consideration both aspects, we compute the (global) walk distance averaged over all 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). Fig. 6.8b 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 station 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 choosing the policy with $w = 1$, customers in 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 to the pole. Therefore, one must be cautious when weighting these results and designing the return policies which impact the customers.

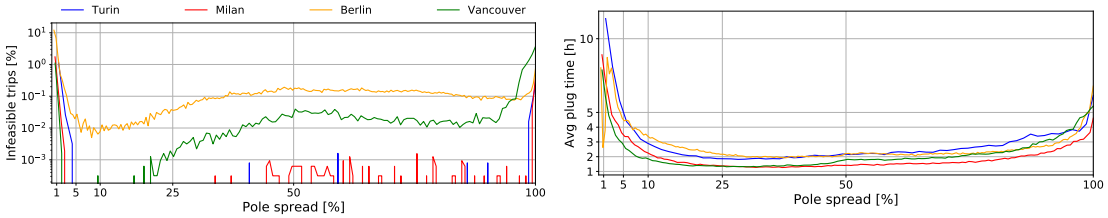
[8] The noise for large N is due to the very small number of rerouting events.



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

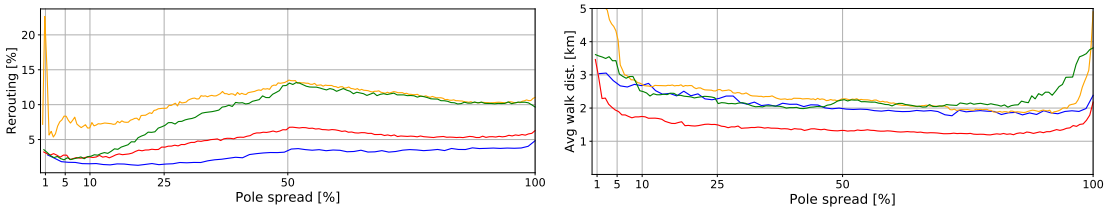
(b) Global average walk distance - the lower the better.

Figure 6.8: 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) Reroute events percentage - the lower the better.

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

Figure 6.9: Impact of pole distribution among zones. Concentrating all poles in very few hubs performs poorly, as well as placing single pole charging stations.

6.8 How to distribute poles in stations

In the previous sections, we have assumed charging stations with $k = 4$ poles each. Here, we study the impact of installing charging station with a larger or smaller number of poles each. We 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, we 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

($N = K, k = 1$). We 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 we pick a constant number of charging poles K, corresponding to a percentage of 7% of charging zones when $k = 4$.[9] Then, we distribute the poles evenly among different numbers of N zones, chosen according to the highest number of parkings (as in the previous Sections). We consider the Hybrid policy with $w = 0.5$ and simulate the resulting system.

From top to bottom, Fig. 6.9 reports percentage of infeasible trips, the average time cars are plugged to a charging pole (even if they are completely charged), the percentage of rerouted trips, and the average walk distance for rerouted 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 Fig. 6.9a. 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 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 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 Fig. 6.9b.

Fig. 6.9c shows the percentage of reroutings. Here we also observe that it is better to have a low spread percentage. For the region where we have negligible infeasible trips (spread between 5 and 20), the percentage of reroutings increase both because poles start to be located in less popular destination, and because cars are charged for less time (see Fig. 6.9b).

Lastly, we show walk distance for rerouted trips in Fig. 6.9d. As expected, the walk distance slowly decreases with spread percentage. Indeed, when customers are rerouted, 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 rented and returned, so to increase the chance to find a free pole, and let the battery

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

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. While 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.

6.9 Conclusion

Designing an electric vehicle free floating car sharing system leads to many interesting problems and trade-offs. In this work, we 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, we demonstrated that it is better to place charging stations within popular parking area (e.g., downtown), even if parking duration is short.

We have shown that a FFCS solution with electric vehicles can almost sustain itself, even with very few charging stations (8-10% of city zones). These results are obtained also 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 take into account the trade-off between usability, costs and benefits for the customers. Our results hint for possible alternative design solution, i.e., the adoption of simple relocation policies that would move cars that need a charge only, a promising solution to limit discomfort for customers due to rerouting enforcement. This could be achieved by considering giving incentives to customers.

Chapter 7

Charging station placement optmization

7.1 Abstract

In this work we consider the design of a Free Floating Car Sharing (FFCS) system based on Electric Vehicles. We face the problems of finding the optimal placement of charging stations, and the design of smart car return policies, i.e, how many and where to place charging stations, and whether to ask or not customers to return the car to a charging pole.

We leverage actual data containing rentals performed by Car2Go customers. We obtain information for several months worth of actual trips in the city of Turin, our use case. Via trace driven simulations, we replay the exact same trips while simulating electric car based FFCS, to accurately gauge battery discharging and recharging. With this, we compare different charging station placements, also driven by optimisation algorithms. Moreover, we observe the impact of collaborative or selfish car return policies.

Results are surprisingly: just as few as 13 charging stations (52 poles) guarantee a fleet of 377 vehicles running in a 1 million inhabitant city to work flawlessly, with limited customer's discomfort. We believe our data driven methodology helps researchers and car sharing providers discerning different design solutions. For this, we make available all data and tools to foster further studies in these directions.

7.2 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 Jörg Firnkorn and Müller, 2011. 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 Jorg Firnkorn and Muller, 2015. However, the design of a system based on electric cars entails the deployment of a charging station network Falvo et al., 2014.

In this work, we 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 Chen et al., 2013; Jiao et al., 2016; Phonrattanasak and Leeprechanon, 2014.

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 paper we face both the above problems. Notice that the number of charging stations is directly related to system installation costs.

We strongly believe actual usage data is fundamental to answer these questions. While in the past some works have proposed solutions for the design of electric FFCS Jorg Firnkorn and Muller, 2015; Weikl and Bogenberger, 2015, we are among the first to take a complete data-driven approach Chen et al., 2013; Hess et al., 2012; Jiao et al., 2016; Marc et al., 2015; Phonrattanasak and Leeprechanon, 2014; Rickenberg et al., 2013 in an electric FFCS. Here, we extend our preliminary works Michele Cocca, Giordano, Mellia, et al., 2018; Michele Cocca, Giordano, Vassio, et al., 2018 by presenting a more extensive evaluation, and introducing two search algorithms.

We start by describing the system we implemented to collect real data from the FFCS systems currently in use in the city of Turin (Italy), which we consider as a test case. Despite being based on internal combustion engine cars, this data perfectly captures the actual usage patterns of regular customers **UMAP**. This naturally factors the desired origin and destination of trips and the time varying demand, including special events such as sport matches or strikes.

We next leverage the data collected from more than 2 months of rentals in 2017. First, we characterise how customers actually use the FFCS, in terms of rental/parking event duration, and of the origin/destination of hundred of thousand trips. Then, we develop a flexible event driven simulator that replays 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, we run thorough simulations to understand the implication of the design choices, such as charging station placement algorithms, and car return policies.

Results show 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.

Next we 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 Flath et al., 2012 proposed. This kind of return policies are inspired by the user-relocation model presented in A. Brendel, Lichtenberg, et al., 2017; A. Brendel, Tim Brennecke, et al., n.d.; A. B. Brendel et al., 2017; Sebastian Wagner et al., 2015, where the relocation is driven by the presence of parking and charging stations and not by the demand areas. We 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, we further optimise the placement of the charging station by means of global optimisation algorithms. We 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.

We 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. Our 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. In an effort to allow reproducibility and extend our results, we make both the data set and the simulator publicly available as open source *UMAP and electric car sharing simulator. Anonymized datasaset of 2 months of trips of car sharing users in the city of Turin* n.d.

After discussing related work in Section 7.3, we present our methodology to collect data and characterise our data-set in Section 7.4. In Section 7.5 we describe the simulation model, its parameters and metrics of interest. Section 7.6 presents the different placement heuristics and the algorithms we design to optimise the placement. Section 7.7 discusses the impact of simple charging stations placement policies and return policies, while Section 7.8 reports the results ofthe optimisation and their validation. Section 7.10 discusses limitations and future work, before drawing conclusions in Section 7.11.

7.3 Related work

The diffusion of the free floating approach to car sharing led to an increasing attention by many researchers, with many analyses of these systems and their extension to electrical vehicles. The studies performed in 2011 by Finkorn and Müller Jörg Firnkorn and Müller, 2011, 2012 are the first attempts to analyse benefits of FFCS for the population. Their results on customers' characterisation, like travelled distances and rental duration, are similar to ours. Later works Habibi et al., 2016; Kortum et al., 2016; Schmöller et al., 2015 also collected data and analysed 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. We also introduced UMAP **UMAP** - a system to harvest data by crawling FFCS websites. Here we use traces collected with UMAP to drive our 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 Jorg Firnkorn and Muller, 2015 investigated the positive influence and feasibility of an electric FFCS systems. Authors in Hess et al., 2012 show the benefits of placing charging stations with different capacity according to the car parking duration. Authors of Bi et al., 2017 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 minimising cost of installation, power loss and maintenance Michele Cocca, Giordano, Vassio, et al., 2018; Jiao et al., 2016; Phonrattanasak and Leeprechanon, 2014, or by minimising the customers' walked distances necessary to reach a charging pole Chen et al., 2013. In Phonrattanasak and Leeprechanon, 2014, authors study the impact on the power distribution grid, with limited focus on FFCS performance. Authors of Jiao et al., 2016 instead focus on charging station design to minimise customers' anxiety. In our previous work Michele Cocca, Giordano, Vassio, et al., 2018, we presented a study of charging station placement based on actual data. Here we build upon this work, and present a more in depth study that includes global optimisation algorithms, never considered before.

Other works focus on station-based car sharing systems. Authors of Rickenberg et al., 2013 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 Marc et al., 2015. Here they use synthetic data and other socio-economic information to estimate the demand. Both works are similar to our in spirit, but are limited to station-based car sharing.

Considering return policies, an interesting data driven research is presented in Flath et al., 2012. The authors focus on 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 to a minimum threshold. This is similar to the return policies we consider in this paper.

Other works focus on FFCS with EVs to study the revenue considering a demand-supply scenario for energy Eisel et al., 2015, introducing policies to free charging stations when occupied by fully charged cars Weikl and Bogenberger, 2015, maximising revenue by moving cars in areas of high-demand Sebastian Wagner et al., 2015, or providing incentives to customers to balance fleet A. Brendel, Lichtenberg, et al., 2017; A. Brendel, Tim Brennecke, et al., n.d.; A. B. Brendel et al., 2017. This works are orthogonal to our.

To the best of our knowledge, we are the first to take a completely data driven approach for designing an electric FFCS system by optimising different metrics impacting customer experience.

7.4 Data collection and characterisation

Obtaining mobility data is fundamental in the design of transport systems. Nowadays, the diffusion of ICT technologies makes collecting actual usage data much simpler. Here, we present our approach to collect data from currently running FFCSs. For this, we designed and engineered *UMAP* **UMAP**. It is composed by three modules: a data acquisition, a data normalisation and integration, and a data characterisation and analysis module. *UMAP* is freely accessible as open source at *UMAP and electric car sharing simulator. Anonymized datasaset of 2 months of trips of car sharing users in the city of Turin* n.d.

7.4.1 Data Acquisition

The first step consists in the data acquisition from the FFCS platform of interest. Each platform exposes information through a web-service that lets customers find positions of cars when available for rental. *UMAP* implements crawler-modules that harvests these web-service to collect, at periodic time instant, which cars are available. For now, we have developed crawlers for two FFCS providers, Car2Go and Enjoy, both offering services in Italy. For Car2Go, we rely on their public APIs [1], while for Enjoy we implemented a custom crawler by reverse engineering API. In both cases, the system returns the currently available cars using a JSON document, for each city in which they are present. *UMAP* takes periodic *snapshots* describing which cars are parked and ready for rental.

In a nutshell, a car is described as an object annotated by information like plate,

[1] Car2Go API, `https://www.car2go.com/api/tou.htm`, service subject to approval by Car2Go. Approval granted in September 2016, service disconnected in January 2018.

vehicle identification number (VIN), fuel level, model, parked location, etc. This is instrumental 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. When a customer reserves and rents it, the object disappears, and reappears later when the customer ends the rental, likely in a different location.

UMAP takes a snapshot $S(t)$ every minute, a reasonable time resolution to capture car rental and return events, while avoiding overloading the servers. In particular, we extract the *VIN* or *plate* field to uniquely identify a car, the parking *coordinates* reported with *longitude* and *latitude* by the in-car GPS, and the *fuel* level of the car.[2]

We started collecting data with *UMAP* in December 2016, and we have collected more than a year of data in all the 22 and 5 cities where Car2Go and Enjoy offer service, respectively.

7.4.2 Data Normalisation and Integration

In this second module we process and consolidate each snapshot $S(t)$ to obtain *booking* and *parking* periods for each car. The intuition is to track the availability of each car, and rebuild the history of parking and booking periods over time: when some customer books a car at time $[t-1, t)$, the latter "disappears" from the system starting from $S(t)$. We identify this events, and record it by creating a new booking, and setting the start time ($t_s = t$) and the start position. When the customer ends the booking, the car "reappears" in the system in $S(t2), t2 > t$. We record this event, by setting the end time ($t_e = t2$) and end position of the booking. After that, for the same car, a new parking period starts, hence we record a new parking event with its initial time ($t_s = t2$) and its position. When the car "disappears" again in $S(t3), t3 > t2$, we record end time ($t_e = t3$) of the parking event, and we start recording a new booking.

Next, we define a procedure to filter bookings and to obtain actual *rentals*. The FFCS allows customers to reserve a car prior the actual rental starts. In case the customer cancels the reservation, the car becomes available, creating a booking in our data-set, for which no rental actually ever happened. Conversely, some bookings last for several days or weeks, e.g., due to a car going offline, or under repair. At last, the back-end system may sometimes fail, generating spurious bookings. *UMAP* filters these artefacts to obtain actual rentals: Initial and final positions must be at least 500 m far apart, booking duration must be greater than 3 minutes, and shorter than 1 hour.

At last, we need to estimate the possible driving path and length from the knowledge of only the initial and final position. Euclidean distance between starting and ending coordinates of the trip represents a lower bound of the real driven distance, since cars have to follow the topology of the city and traffic laws. To estimate the real driving

[2]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.

distance, we apply a corrective factor, that we obtain again from data. In more details, given a rental, we use the information returned by the Google Directions API related to the best driving path from the initial to the final position of the rental. [3] Then, we compute the ratio between the returned driving distance, and the euclidean distance. We collect this information at time the trip end is observed.[4] We repeat this for about 10 000 trips, observing the distribution of the ratio which ranges between 1 and 2, with most of values around the median value of 1.4. We use the latter value as corrective factor to obtain the driving distance from the euclidean distance.

In this work, we consider 8 weeks between September and November 2017 for the Car2Go provider to optimise the charging station placement. We collected about 190 000 bookings that generated 125 000 actual rentals from the 377 cars of the Car2Go fleet in Turin. To verify the robustness of the solution, we use other two traces collected in June-July 2017 and December 2017-January 2018.

Our data takes into account all the rentals, also in presence of special events generating a higher demand (like strikes, sport events, concerts, etc.). Therefore our data captures both normal daily traffic patterns, and anomalies.

In the following, we provide a quick characterisation of the data, which is instrumental to guide the design of the charging station placement algorithms.

7.4.3 Data characterisation

First focus on the characterisation of the (estimated) distance travelled during rentals. This figure is interesting since it is directly related to the minimum amount of charge an electric car shall have to complete a trip. Fig. 7.1a shows the empirical Cumulative Density Function (CDF) of the distances covered over all rentals. 97% of the trips covers less than 10 km, roughly corresponding to the operative area diameter in Turin. The remaining 3% are trips to and from the airport, up to 19 km far away.[5]

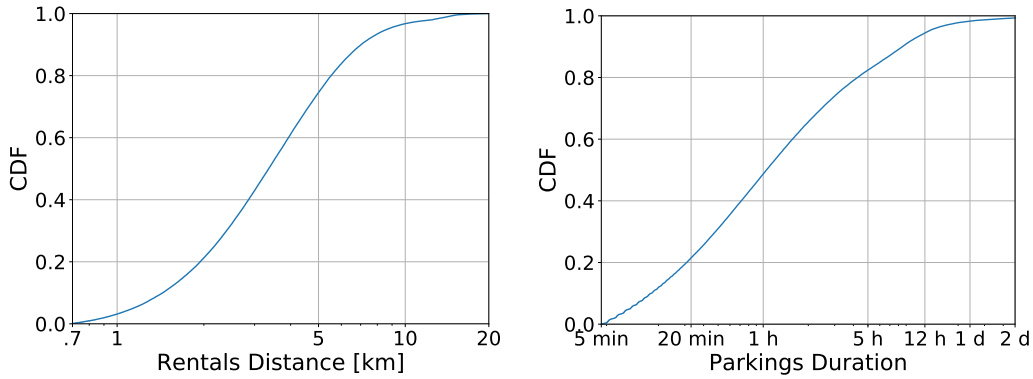
Next, we investigate the parking duration. This figure is interesting since it is related to the amount of charge a parked car obtains when attached to a charging station. Fig. 7.1b illustrates the empirical CDF of parking duration. Interestingly, more than half of the parking events lasts less than 1 hour. This is due to the high utilisation of cars in FFCS, especially during business hours. Conversely, 10% of the parking events lasts more than 8 hours, with some cars that are left parked for days. The former are probably due to overnight parkings, while the latter hint for some cars parked in areas with few customers.

---

[3] https://developers.google.com/maps/documentation/distance-matrix/, freely available for a limited number of queries.

[4] Google Directions API factor eventual traffic congestion at different time of the day. As such, we include its impact in our model.

[5] The Turin airport has a reserved parking for Car2Go vehicles. It is about 15 km far away from the city centre. Distances for trips to the airport, reached by a straight highway, are not corrected.



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

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

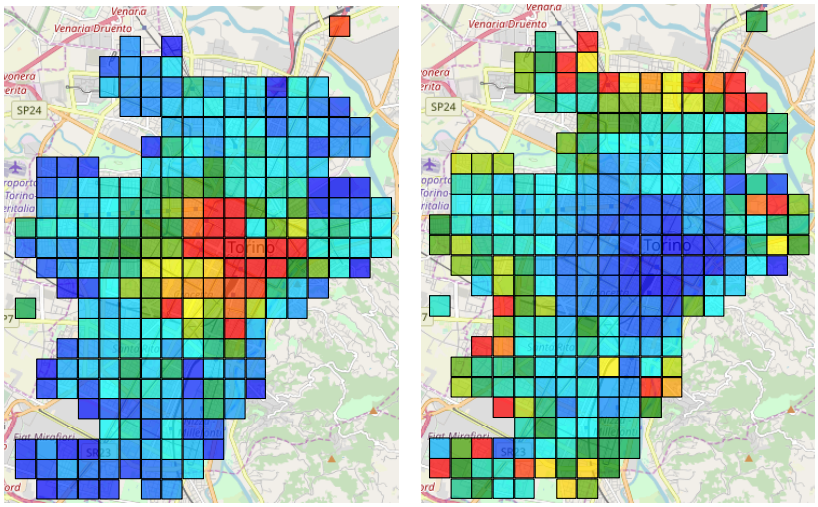
Figure 7.1: Characteristics of the trips in our data-set.

Next, we analyse how parking habits are different in the city area. For this, we divide the service operative area using a grid of squared zones of 500x500 meters, obtaining 261 zones covering Turin. For each zone, we compute the total number of parkings recorded and the average parking time.

Fig. 7.2a shows the heatmap of the total number of parkings in each city zone. The warmer the colour is, the more frequently cars are parked here. The hot areas correspond to the city centre which exhibits the highest number of parkings. Customers rely on car sharing for travelling 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. On the contrary, few parkings are observed in the suburbs (down to less than 1 event per day), where people likely return home in the evening **UMAP**.

Fig. 7.2b shows the heatmap of the average parking time for each zone. Peaks are on borders of the operative area, where parking events last more than 24 hours. Few cars reach these peripheries and rest unused for 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.

The large spread of parking density and duration challenges the decision on where to place charging stations. Indeed, if placed in areas where cars are frequently parked but for short time (e.g., city centre), batteries would get little charge. If placed in areas where few cars stay parked for long time (e.g., suburbs), cars will be fully charged cars but occupying the station for long time.



(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.

7.5 Electric car sharing simulator

Our 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.

We simulate a fixed fleet of electric cars. Each car is characterised by its parking location, and the current status of battery charge. The simulator takes as input the pre-recorded data-set of rentals, i.e., the trace, characterised by the start and end time, and initial and final geographic coordinates. For simplicity, space is divided into 261 zones of 500 x 500 m each (as explained in Section 7.4).

7.5.1 Trace event processing

Each recorded rental reflects a mobility interest of a customer, i.e., a desired trip. In more details, each trip $i \in \mathcal{I}$ is characterised by its start and end time, $t_s(i)$ and $t_e(i)$, and origin and destination coordinates, $o(i)$ and $d(i)$. We associate each position to the zone $O(i) = zone(o(i))$ and $D(i) = zone(d(i))$. We assume a charging station cs, composed of k poles, can be placed at the centre of a given zone $z \in \mathcal{Z}$, so either $cs(z) = 1$ if the station is present, or $cs(z) = 0$ otherwise. $N = \sum_{z \in \mathcal{Z}} cs(z)$ is the total number of zones equipped with charging stations, with $K = N \cdot k$ the total number of poles.

We have a set $\mathcal{A}$ of cars, where each car $a \in \mathcal{A}$ at time t is characterised by its position $p(a,t)$, its zone $P(a,t) = zone(p(a,t))$, and the residual battery capacity $c(a,t) \in [0, C]$, with C being the maximum nominal capacity.

The simulator processes each rental event i in temporal order. When a car rental start event i at time $t = t_s(i)$ is processed, the (simulated) customer looks for a car in the initial position zone $O(i)$. If cars are present, the customer rents the most charged one, independently whether the car is at a pole being charged or not.[6] In formulas, we get a car $\bar{a} \in \mathscr{A}$ such that:

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

If no car is present, the customer walks to the closest zone containing an available car, mimicking the normal behaviour of FFCS customers that look for the closest car to rent on their smartphones. A car rental end event is then scheduled using the trace final time $t_e(i)$ and desired destination location $d(i)$. When car a rental end event at time $t_e(i)$ is processed, the customer returns the car in $p(a, t_e(i))$, chosen according to the behaviour described in the next paragraph. The simulator updates the battery charge status by consuming an amount of power proportional to the trip distance:

$$c(a, t_e(i)) = \max\left(c(a, t_s(i)) - Energy(p(a, t_s(i)), p(a, t_e(i)))\right)$$

with $Energy(\cdot)$ that models the energy necessary to go from the car origin $p(a, t_s(i))$ to the car destination $p(a, t_e(i))$. Here we consider $Energy(\cdot)$ to be dependent only on the two positions and proportional to their distance, but more complicated functions can be easily implemented.

In case the battery level drops below 0 ($c(a, t_e(i)) = 0$), the trip i is declared *infeasible*. The discharged car still performs further trips, all marked as infeasible, until it reaches a charging station.[7]

Depending from the return policy, the customer may connect the car to a charging pole. We investigate the following return policies:

- *Free Floating*: the customer opportunistically connects the car to a charging pole if and only if it is available (present and free) in the final desired zone $D(i)$;
- *Needed*: cars are connected to a pole only when the battery level at the end of the rental goes below a minimum percentage threshold α, i.e., $(c(a, t_s(i)) - Energy(p(a, t_s(i)), d(i)))/C \leq \alpha$. This implies the customer can be *rerouted* to the closest zone with an available free charging pole, if none exists in the desired final zone $d(i)$;
- *Hybrid*: the customers follow the Needed policy, but voluntarily connect to a charging pole – if available – in the desired ending zone, whatever car charge status is;

[6] We choose this policy because people are worried about vehicle range Jung et al., 2015.

[7] This is instrumental to give an exhausted car the chance to recover energy.

The *Free Floating* policy never obliges the customer to bring the car far from the desired ending location, even in case battery charge is close to exhaustion. *Needed* mandates to connect cars to a charge station only if 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.

Notice that policies similar to *Needed* have been introduced in Flath et al., 2012, 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.

7.5.2 Performance metrics and parameters

We measure the following metrics, that we identify having influence in 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*: walked distance from the desired destination. This is considered non-zero both when the car is charged or rerouted. The walk distance when returning the car to a pole in the desired final destination is considered to be 150 m, i.e., the average distance from any point to the centre of a square of 500 m side;

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

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

We 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 travelled distance (12.9 $kWh/100\,km$).[8] 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. We

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

model a simple linear charge profile (complete charge in 8 hours and 50 minutes in our case). For *Hybrid* and *Needed* policy we set the minimum battery charge threshold, α, equal to 25%. This is a precautionary approach, since the maximum travel distance is 19 km (Fig. 7.1a), corresponding to about 14% of the battery capacity. At last, the initial position of the cars, only affecting the initial transient, is randomly chosen.

Our Python simulator completes a single simulation including 125 000 rentals in less than 5 seconds.[9] To post-process generated results and extract aggregated data, we use PySpark[10] on a Big Data cluster of 30 nodes.

7.6 Strategies for charging station placement

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

7.6.1 Heuristic placements

Each zone z is assigned a likelihood l_z. We greedily choose the top Z zones, according to three likelihood definitions:

- *Average parking time*: l_z is the average parking duration in z as measured in the trace; the stations will be located in the areas where cars stay parked the longest time, hence with more probability to fill-up the battery during a charging event;
- *Total number of parkings*: l_z is the total number of parking events recorded in i in the trace; hence the stations will be located in the areas with more parking events. Likely, more cars will be charged;
- *Random placement*: l_z is an independent and identical distributed random uniform variable. As such, recharging stations result placed at random over the city area;

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

[9] We are able to run up to 40 simulations in parallel on a 40-core Intel Xeon processor with 128 GB of RAM, running Ubuntu 16.04 OS.

[10] http://spark.apache.org/docs/latest/api/python/#

7.6.2 Simulation based advanced optimisation strategies

Given the complexity of the optimisation problem and the humongous space of possible solution, we investigate the adoption of meta-heuristics, a class of global optimisation algorithms Rao, 2009. These algorithms explore the space of possible solutions in smart ways, looking for better solutions while avoiding getting trapped into local minima.

In our 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 we have limited resources in the choice of meta-heuristics to consider. The class of optimisation 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 Floudas and Pardalos, 2006; Rao, 2009. In the literature, there are several architectures and algorithms suitable for global optimisation in tough numerical problems Floudas and Pardalos, 2006, with direct-search class algorithms that explicitly target expensive-optimisation problems.

In our work, we consider a simple local-search algorithm based on a hill-climb method, and a more complex and powerful genetic algorithm. We explicitly design both algorithms with the perspective of reducing the number of simulations. Both are implemented in our open-source tools *UMAP and electric car sharing simulator. Anonymized datasaset of 2 months of trips of car sharing users in the city of Turin* n.d.

We 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 minimised first. Secondly, the algorithms starts minimising the *WalkedDistance%*. In a nutshell, we look for solutions that make all trips feasible, and only then we target the customers discomfort, i.e., reducing the walked distance. As we will better show in the next section, 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 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.

Our hill-climbing algorithm is very similar to the coordinate descent version Wright, 2015. We start from the best configuration found among the three heuristics. At each iteration, the algorithm randomly picks a charging station, and moves it in a empty neighbouring zone, i.e., north, south, east and west adjacent zones. All other charging stations are left untouched. If there is a direction of improvement, we perform a line search along the best direction, i.e., we keep moving the same station along the same direction. When no improvement is possible, we take another charging station at random, and try to move it as before. When no improvement is found after a complete cycle of all charging station, a local minimum is reached and the algorithm exits. We also stop the local search after a maximum number of visited solutions.

In our implementation, we check multiple neighbours in parallel. Moreover, the algorithm keeps memory of all tested configurations, hence avoiding useless and expensive simulations. For details see *UMAP and electric car sharing simulator. Anonymized datasaset of 2 months of trips of car sharing users in the city of Turin* n.d. In our experiments, on average the optimisation reaches convergence within 1 500 maximum tested solutions.

Genetic optimiser

Genetic algorithms are a particular class of evolutionary algorithms inspired by the natural evolution Goldberg, 1989. They are known to work well when dealing with discrete variable functions, as in our case.

Genetic algorithms start creating a random population of a given number of individuals. In our case, each individual corresponds to a 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 our 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 optimisation loop continues until the maximum number of generations is reached (200 in our 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, we increase the population genetic diversity by increasing the mutation probability ($P\{mutation\} = 0.2$

in this case). This randomises the evolution. More details about the algorithm can be found in our open-source implementation *UMAP and electric car sharing simulator. Anonymized datasaset of 2 months of trips of car sharing users in the city of Turin* n.d.

The algorithm is amenably suitable for parallel implementation since the fitness of each individual belonging to a specific generation can be analysed 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 limited portion of the space.

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

7.7 Impact of heuristic placements and return policies

7.7.1 Impact of heuristic charging station placements

In this first set of experiments we evaluate which heuristic placement performs better. Firstly, for each heuristic we consider the simple opportunistic *Free Floating* car return policy, i.e., customers always return the car in the desired zone, and connect it to a charging pole if available. The goal is to check if this simple return mechanism is sustainable with electric cars, and gauge the impact of the three different heuristics for the charging station placement.

Fig. 7.3 shows the performance in terms of infeasible trip percentage versus an increasing percentage of charging zones, ranging from 1% to 30%. In Turin, this corresponds to install from 2 to 78 charging stations overall (reported in the top x-axis). We observe notably different performances. First, the average parking time placement policy (*Avg time*) performs the worst, with still 6% of infeasible trips even when more than 25% of zones are equipped with charging stations. Even a simple random placement performs better (*Mean rnd*, obtained as the average of 10 independent instances). The total number of parkings placement (*Num parking*) reaches about 2% of infeasible trips at 10% coverage, reaching self-sustainability when more than 20% of zones are equipped (no infeasible trips).

The intuition of why such a striking difference is given by the different properties of areas where the heuristics place charging stations. *Avg time* placement favours peripheral zones where few trips ends, and where cars stay parked for long time (see Fig. 7.2b). On the contrary, *Num parking* favours city centre areas, where cars frequently are parked for short time (see Fig. 7.2a). Indeed, in the whole simulation, for $Z = 40$ ($\approx$ 15% of zones), only 7 430 charges have been recorded for *Avg time*, compared with 47 628 charges of the *Num parking*. Even if plugged time is shorter, the *Num parking* policy allows the cars to charge the (little) energy consumed during the (short) trips.

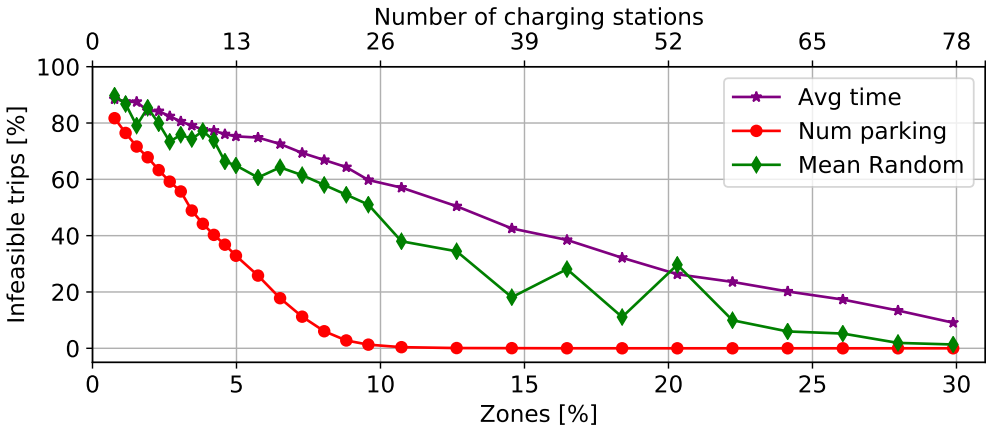


Figure 7.3: Percentage of unfeasible trips as function of charging stations (percentage and number of city zones), for the different heuristic placements.

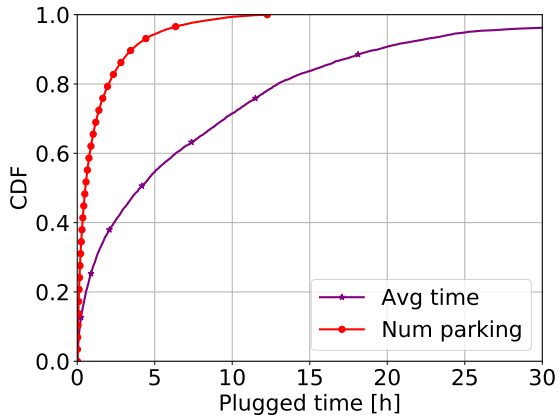


Figure 7.4: CDF of the time spent by a car at a charging station (Z=40), for *Num parking* and *Avg time* placement algorithms.

Moreover, as shown in Fig. 7.4 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 to use those poles and increasing the number of infeasible trips.

In a nutshell the best approach among these three heuristics is to place charging stations in the central areas, in which the parkings last less but are more frequent. For this reason, we will use the *Num parking* placement algorithm as best heuristic for the remaining of the paper.

7.7.2 Impact of return policy

We now investigate the impact of the three different return policies. We quantify the implications of forcing customers to return the car to a different zone than the desired one, when the battery is below a critical level (i.e., below the percentage threshold $\alpha = 25\%$ of full battery capacity).

We focus again on the infeasible trip percentage with respect to the charging station coverage. Fig. 7.5 shows results, with *Num Parking* placement. *Needed* and *Hybrid* policies perform much better than the opportunistic *Free Floating*. In details, *Hybrid* and *Needed* guarantee to successfully conclude all trips with just $Z = 11$ and $Z = 15$ charging zones, respectively (4.2% and 5% of the zones), while *Free Floating* reaches this goal only at $Z = 60$ (23% of zones). In a nutshell, adopting a policy which mandates customers to charge the cars when battery level gets low drastically reduces the number of infeasible trips, even with a handful of charging stations.

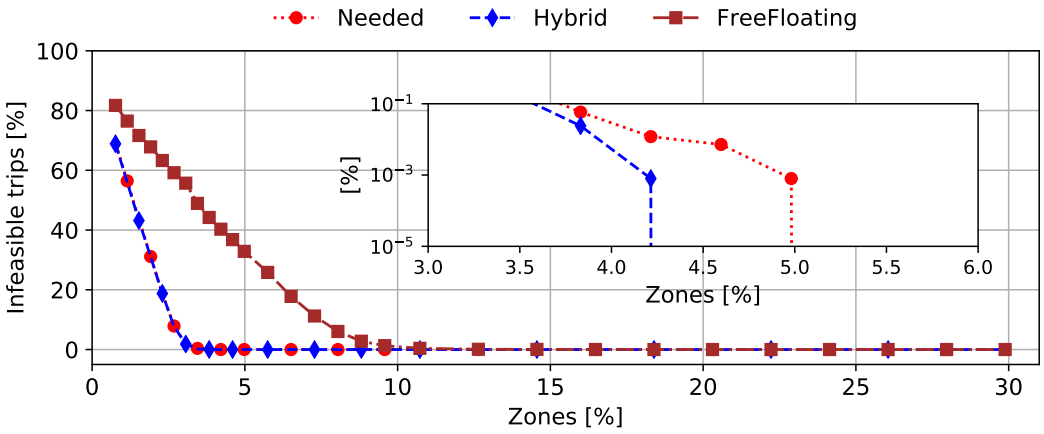
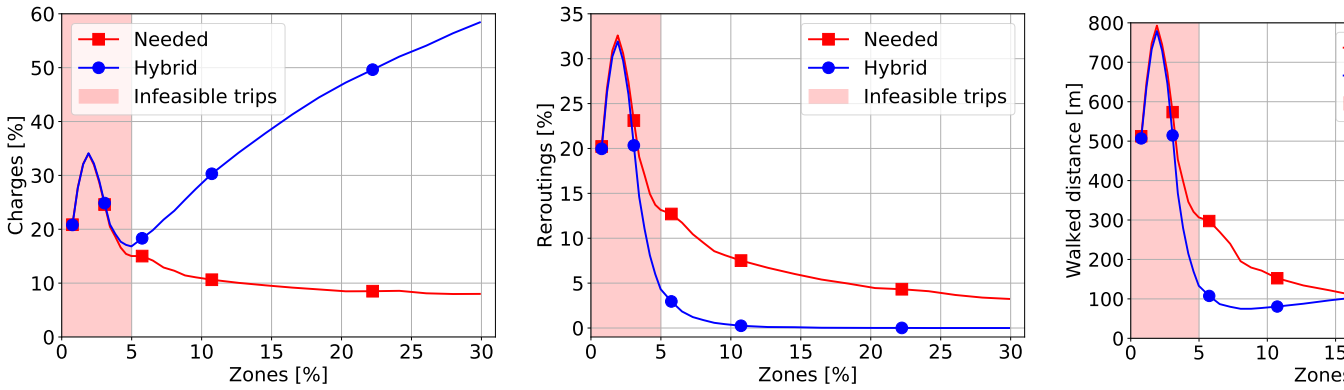


Figure 7.5: Percentage of infeasible trips for different zone coverage percentage analysing the return policies. The inset highlights where the infeasible trips go to 0.

Customers' discomfort

Forcing customers to charge batteries, even at cost or rerouting them has clearly an impact on their comfort. Indeed, forcing a customer to park in a charging station can be annoying, because of the burden of reach the charging station, and losing time to plug to and unplug the car from the pole. Even worse, rerouting customers to the closest zone for charging increases the distances they have to walk to reach the desired destination.

Fig. 7.6a reports the percentage of trips that end by charging the car. Shaded area highlights the infeasible region, where the lack of charging zones create artefacts. Focusing on the feasible region instead, the two curves start with similar values, but then



(a) Percentage of trips ending with a charge. (b) Rerouted trips percentage. (c) Walked distance, aver

Figure 7.6: Metrics of interests when comparing Hybrid and Needed return policy (Max parking placement adopted).

they diverge. Interestingly, the percentage of charges decreases for the *Needed* policy, getting as low as 8%. This suggests a better usage of each single charge, i.e., the battery fills up and does not need frequent charges. Conversely, the *Hybrid* policy increases steadily the fraction of trips where customers have to plug to a pole. This because the higher Z, the higher the probability of ending the trip in a zone with a free charging pole.

Fig. 7.6b represents the rerouting percentage in function of charging stations coverage. Rerouting probability decreases as expected: the more the stations are, the more likely customers find a charging station at their desired final zone. Yet, the two policies have different performance. The *Hybrid* policy is less likely to reroute customers. In fact, by opportunistically connecting the car to a charging pole if available, the average battery charge is high, thus decreasing the rerouting probability. With more than 7% of charging zones, the percentage of rerouting is already lower than 1% for *Hybrid* policy. In a nutshell, *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 to the pole. Therefore, one must be cautious when weighting these results and designing return policies which impact the customers' comfort.

Whenever the system forces a customer to park the car in a charging station, customers may be routed far from their desired destination. If the charging station is not in the final destination zone, the customer has to walk by at least 500 m to reach their final destination. Even in case the charging station is found inside the final zone, the customer will have to park at the charging station, instead of the desired destination. For this reason, we evaluate the average distance the customers have to walk to reach their actual final destination. In details, when the customers suffer rerouting, we evaluate the actual distance between the charging station and the final destination. When

they end in a charging zone and plug the car, we count an average distance of 150 m. Finally, if the customers do not charge the car, we assume they arrived at their final destination directly.

Results are shown in Fig. 7.6c. Consider the feasible region. The *Needed* policy exhibits a decreasing trend (from 280 m to 60 m). On the contrary, the *Hybrid* policy first exhibits a decrease (minimum of 50 m at 8% of charging zones), but then it slowly increases till it overtakes the *Needed* policy. This is due to the fact that with few charging stations (6-15%) the number of charges is limited ($<$ 30%, see Fig. 7.6a) by the availability of charging stations. Instead, when this number grow, the opportunistic *Hybrid* policy forces the customer to walk more times within the ending zone. To this extent, the *Needed* policy performs better.

Even if the values of walked distances seem very small, remember that they are averaged over all trips. Restricting to the few long walks due to rerouting (not reported for the sake of brevity), the walked distance that we observe ranges from 2 200 m to 1 500 m, with similar behaviours for the two policies. Recall that the charging station placement algorithm likely places stations mainly in the city centre. Therefore, the charging stations are concentrated in a small area, so that rerouting from the suburbs significantly affects the walked distance when rerouted. This gives hope for further optimisation, as we will see in the next section.

Given the very few rerouting of the *Hybrid policy*, one can envision a system which directly takes care of those very few cars that need a battery charge, i.e. by relocating vehicles. For instance less than 3 cars per day would need to be relocated with 15% coverage or higher.

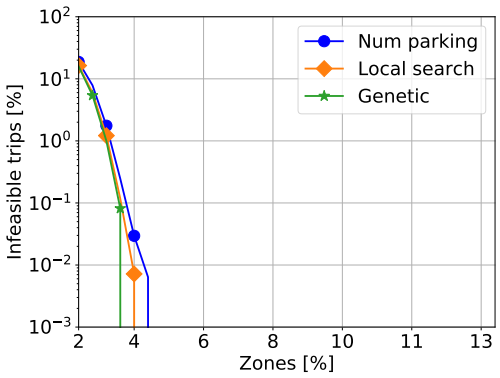
In summary, with *Hybrid* policy, less than 15% of zones guarantees all trips to be feasible, reduces the walked distance, asks for few rerouting events, at the cost of moderately high percentage of times (40%) customers are asked to charge the battery.

7.8 Meta-heuristic optimisation of the charging station placement

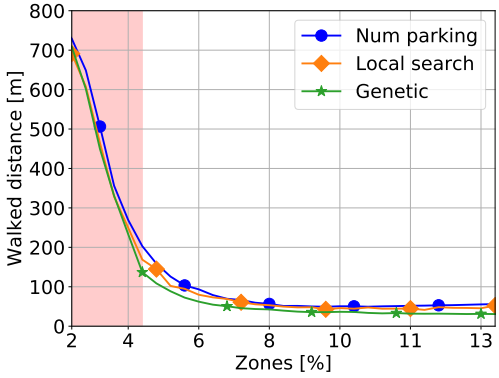
In the previous section, we saw how the *Num Parking* placement heuristic works better than the other two. Weighting also charges and rerouting, the *Hybrid* policy shows better performances than *Needed*. For this reason, in this section, we focus on the *Hybrid* return policy with charging stations covering less than 15% of the zones. We further optimise 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 7.9, 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 guarantee 0 infeasibile trips, and to minimise

the overall distance the customer has to walk to reach the final destination.

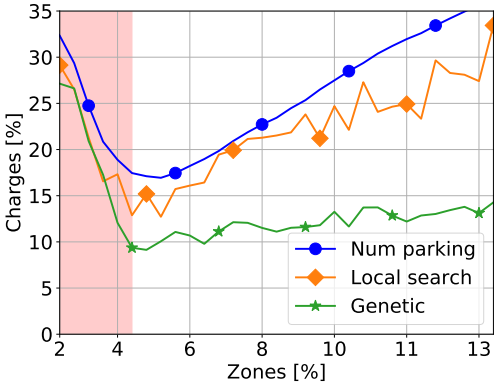


(a) Percentage of infeasible trips. Y-Axis is logarithmic.

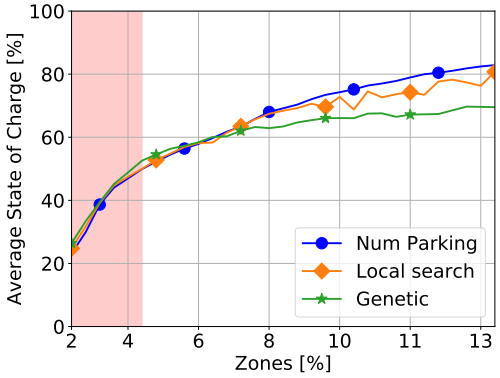


(b) Average walked distance.

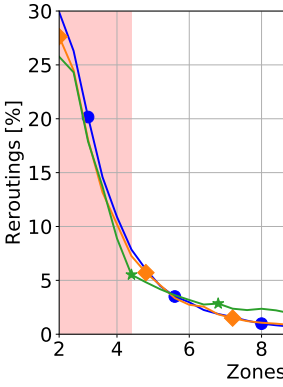
Figure 7.7: Objective metrics to minimise in the optimisation - with *Hybrid* return policy.



(a) Percentage of trips ending with a charge.



(b) Average state of charge.



(c) Rerouted trips

Figure 7.8: *Genetic* and *Local Search* optimisation results for metrics of interests (*Hybrid* return policy adopted).

Fig. ?? reports the two target metrics, for all the optimised configurations. Firstly, in Fig. 7.7a we 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.

Fig. 7.7b 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 Fig. 7.8 we further study the new solutions on other metrics. Fig. 7.8a 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 minimising the walked distance, we 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 Fig. 7.8b, which reports the average state of charge of the car battery. No major difference 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.

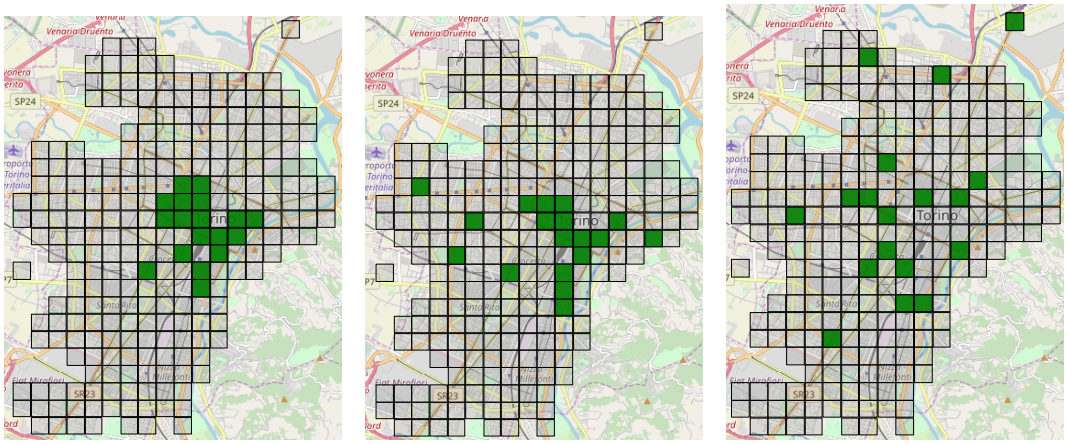
Consider next Fig. 7.8c, which details the percentage of reroutes. We can see how the trend is the opposite with respect to the previous ones. Here, the *Genetic* optimised solution show a little higher rerouting percentage than the *Num parking*. In particular, the *Genetic* algorithm reaches 1.3% of reroutes, while *Num parking* decreases down to 0.2% of reroutes. Indeed, the *Genetic* algorithm places charging stations not only where most rentals ends, 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 Fig. 7.8d we 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.

7.8.1 Charging station placement visualisation

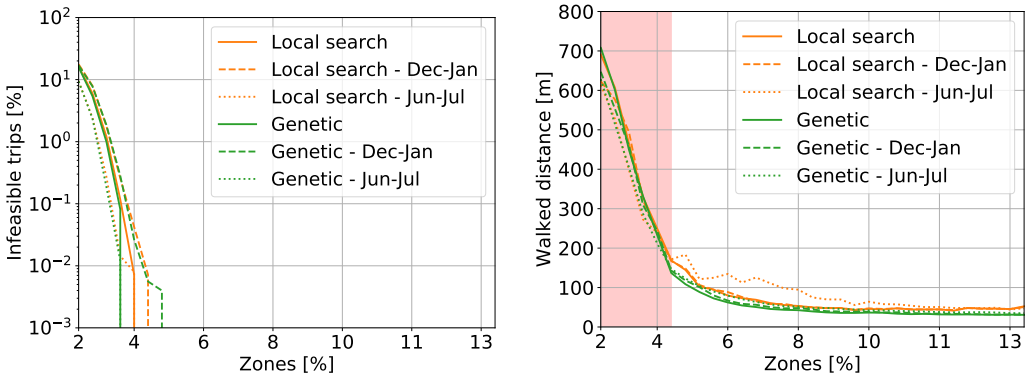
To give a feeling about the differences in the solutions found by different algorithms, Fig. 7.9 reports the solutions obtained with 7% of the zones equipped with charging stations (i.e., $Z = 18$ zones).

Num parking solution, Fig. 7.9a, places most of the charging stations in downtown area and near the main train stations. *Local Search*, Fig. 7.9b, still has many zones in common with *Num parking*, the solution it started from. It just spreads some charging



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

Figure 7.9: 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.



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

Figure 7.10: Performance of the optimised configuration, tested on other 2 months long data-sets.

station to cover also some remote zones. The *Genetic* algorithm, Fig. 7.9c, 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.

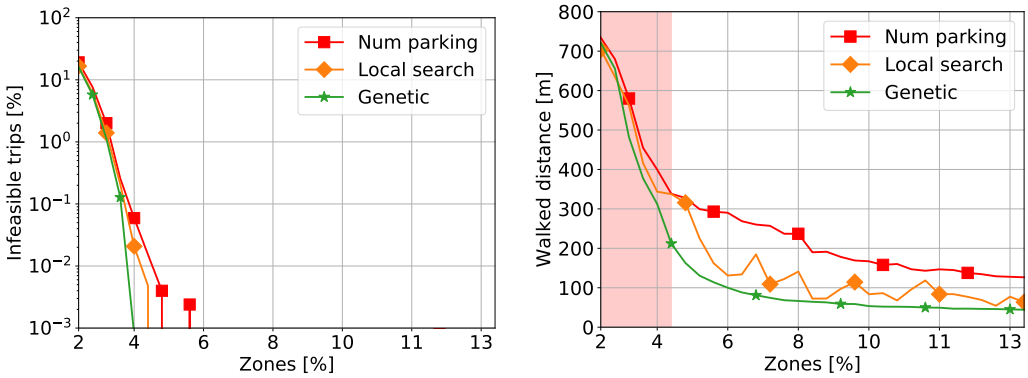
7.8.2 Validation of optimised configurations

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 our findings, we test the output placement configurations by using independent test traces, different from the one used to run the optimisation. We 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. We 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 we 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). We compute the best station placement considering the September and October 2017 trace, and test system performance using the summer and winter traces.

Fig. 7.10 compares results. We 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.

7.9 Placement optimisation for *Needed* return policy

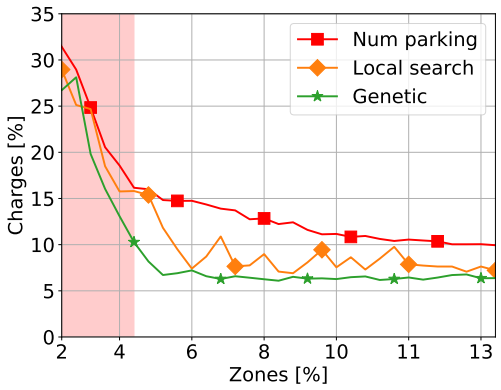


(a) Percentage of infeasible trips. Y-Axis is logarithmic.

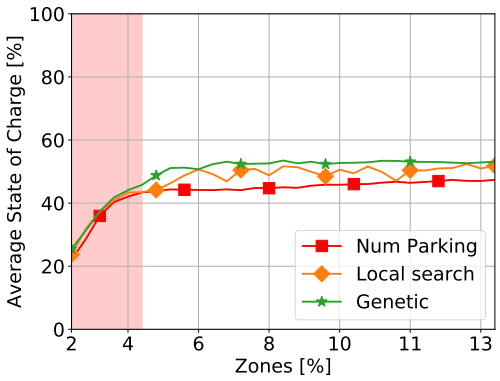
(b) Walked distance, averaged over all trips.

Figure 7.11: Objective metrics to minimise in the optimisation - with *Needed* return policy

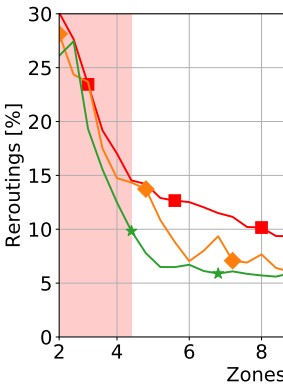
Here we briefly report the results for the optimisation experiments of the *Needed* policy. We followed the same procedure explained in Sec. 7.8 for the *Hybrid* return



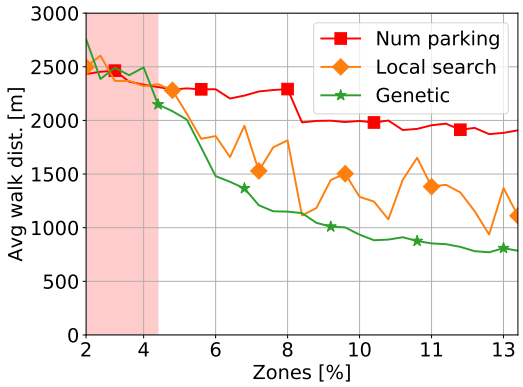
(a) Charges percentage.



(b) Average state of charge.



(c) Rerouted trips



(d) Walked distance when rerouted.

Figure 7.12: *Genetic* and *local search* optimization results for metrics of interests (*Needed* return policy adopted).

policy. As in that case, the genetic algorithm is able to largely optimise the solution, as reported in Fig. 7.11, with *local searches* stuck in local minima. In particular, for the walked distance (Fig. 7.11b), 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.

Fig 7.12 reports the other user discomfort metrics. The two optimisation algorithms reduce the charge events (Fig 7.12a). However, the average state of charge (Fig. 7.12b) is always higher in the optimised solutions than in the *Num parking* configuration. This further demonstrates that a smarter placement allows the car to get more energy for each charge. Fig. 7.12c reports the rerouting percentage for the different algorithms. As expected, the values are larger than with the *Hybrid* policy, since no opportunistic charge is performed. However, the genetic algorithm is able to quickly reach a small value of reroutings, hence better exploiting every charge possibility. Finally, we analyse the distance a user has to walk when rerouted (Fig. 7.12d). 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.

7.10 Discussion and Implication

The mobility model we 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, we briefly discuss two of them.

7.10.1 Scalability

We built the events trace from real rentals in the analysed city. By using this data we 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, we have 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, we can 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 we 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 by UMAP, our methodology can be used to consider the greedy placement of new charging stations on the top of those already present.

7.10.2 Economical Aspects

Economical aspects play too a key role in the placement decision process. In this work, we 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 Smith and Castellano, 2015 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 Sebastian Wagner et al., 2015 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.

Given that, we are currently working on a more complete model to better study the economical impact of electrification by including in the simulator a precise cost model.

7.11 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, we built on actual rental traces to study via accurate simulations the impact of different charging stations placement and charging policies. We considered Turin as a case study, using 2 months of rentals recorded from a currently operational FFCS that we use to run trace driven simulations.

We 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 Our 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 we propose naturally fits the global optimisation algorithm that is able to optimise placement while considering complex customers habits.

We 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. We believe that our approach, based on data and accurate simulations is very promising to design and understand electric FFCS systems in future smart cities, provided actual data is available.

Chapter 8

Geo spatial prediction

8.1 Abstract

Free Floating Car Sharing (FFCS) services are a flexible alternative to car ownership. These transportation services show highly dynamic usage both over different hours of the day, and across different city areas. In this work, we study the problem of predicting FFCS demand patterns – a problem of great importance to an adequate provisioning of the service. We tackle both the prediction of the demand i) over time and ii) over space. We rely on months of real FFCS rides in Vancouver, which constitute our ground truth. We enrich this data with detailed socio-demographic information obtained from large open-data repositories to predict usage patterns. Our aim is to offer a thorough comparison of several machine learning algorithms in terms of accuracy and easiness of training, and to assess the effectiveness of current state-of-art approaches to address the prediction problem. Our results show that it is possible to predict the future usage with relative errors down to 10%, and the spatial prediction can be estimated with relative errors of about 40%. Our study also uncovered the socio-demographic features that most strongly correlate with FFCS usage, providing interesting insights for providers interested into opening service in new regions.

8.2 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 a massive adoption of car sharing service can improve mobility as well as reduce costs and pollution (**12_LitmanCSbenefits**; **14_firnkorn2011will**; **14_firnkorn2012selling**).

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 (Francesco Ciari, Weis, et al., 2016; M. Cocca et al., 2019; Michele Cocca, Giordano, Mellia, et al., 2019).

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 (**UMAP**). 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. Our 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.

Our 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

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

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

Short-Term Memory Neural Network (NN) (Bishop, 2006; Brockwell and Davis, 2016). 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, we correlate socio-demographic indicators with FFCS demand. We 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 data. We map the FFCS demand to Vancouver neighborhoods, and associate them to the socio-demographic data coming from the official Vancouver census. We 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). Our 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 8.3, we describe the data collection methodology we adopt in Section 8.4. Section 8.5 provides a characterization of the datasets, while Section 8.6 and Section 8.7 provide details about the methodologies and results for the temporal and spatial prediction, respectively. Finally, Section 8.8 summarizes our findings.

8.3 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 (Okutani and Stephanedes, 1984) 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 (Clark, 2003) 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, our 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 (**ciari2013estimation**). Later on, as data from operative car sharing platforms became available, researchers started using real data to analyze mobility demand. For instance, previous work (**14_firnkorn2011will**; Catalano et al., 2008) proposed a demand model to forecast the modal split of the urban transport demand. Similarly, other studies (**13_firnkorn2012selling**) 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 (**18_becker2017comparing**) 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 (Francesco Ciari, Weis, et al., 2016) 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 (**schmoller2015empirical**; Schmöller and Bogenberger, 2014) conducted a detailed characterization of a car sharing system in Munich and Berlin. Similarly to our work, they identified features correlated with the demand for shared cars in the target cities. However, our 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 our previous work (Alencar et al., 2019), 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 (M. Cocca et al., 2019). In particular, we tackled the charging station placement problem, showing that the optimal placement requires few stations to satisfy charging requests

in different cities (Michele Cocca, Giordano, Mellia, et al., 2019).

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 (He and Shin, 2019; Hulot et al., 2018), 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 (D. Wang et al., 2017) 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 our prediction problems as well as to have a broader perspective on the challenges involved in car sharing prediction.

8.4 Data gathering methodology

8.4.1 FFCS data collection

We collect data from Car2go, a popular FFCS system that offers its services in more than 25 cities and 3.6 million customers in 2019. In a nutshell, Car2go works as follows. The system knows the position of all cars (available or not) in the fleet. A customer looks for and reserves an available car by using a smartphone application, after which he or she can drive the car. At the end of the ride, the customer parks and returns the car by notifying the FFCS system via the smartphone application. The system records the new position of the car, and makes it available for other customers.

Car2go allows developers to interact with their services through a public Application Programming Interface (API).[3] With this API, we can retrieve the current position of available cars in a given city. Each car is identified by its plate, and it is possible to identify rentals by simply performing periodic queries. In our previous work, we developed a system for collecting data from Car2go's API (**UMAP**). This same system, called Urban Mobility Analysis Platform (UMAP) is used in this work. It allows us to systematically collect precise data about car rentals in all cities where Car2go offers its service. More specifically, UMAP queries the Car2go API every minute to get the currently available cars. It then rebuilds the history of rentals of each car, identifying

[3]The use of the Car2go API (`https://www.car2go.com/api/tou.htm`) is subject to approval by Car2go. We got the approval in September 2016 and continued to collect data until January 2018.

bookings and *parkings*. A *booking* is the time period in which a car is booked by a customer (or in maintenance). Conversely, a *parking* is the time period during which a car has been available for a ride to users.

Since customers can reserve a car and then cancel the reservation afterwards without actually renting it, we consider a *rental* a booking having (i) distance between starting and final locations greater than 500 meters; (ii) travel duration shorter than 1 hour. In a nutshell, we discard those bookings which where not converted into rentals (i.e., when the user reserved the car without actually driving it), and those rentals where the car disappears for long periods (i.e., possibly due to maintenance). We refer the reader to previous work (**UMAP**) for a detailed analysis of these implementation decisions.

Here we focus on Car2go rentals recorded in Vancouver during 10 months of 2017. We chose the city of Vancouver as a case study for two reasons. First, among the cities where Car2Go offers its service Vancouver is the city the highest number of rentals per day. Second, because of the amount of open data made available by the Vancouver municipality. In total, we collect more than 1 million rentals that we use as ground truth to train and test machine learning algorithms to predict service demand across time and space.

8.4.2 Socio-demographic, weather and other open data

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[4] divides the city in 22 official neighborhoods. Our 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.[5] 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);[6] 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.[7] For each of the 22 neighborhoods, we normalize each numerical feature by the

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

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

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

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

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.

8.5 Dataset overview

We 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.

8.5.1 FFCS temporal characterization

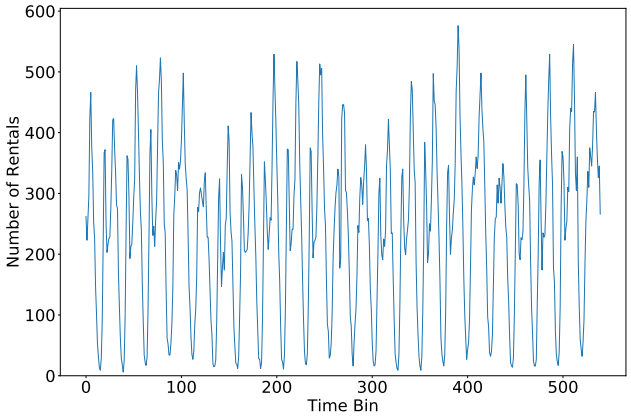
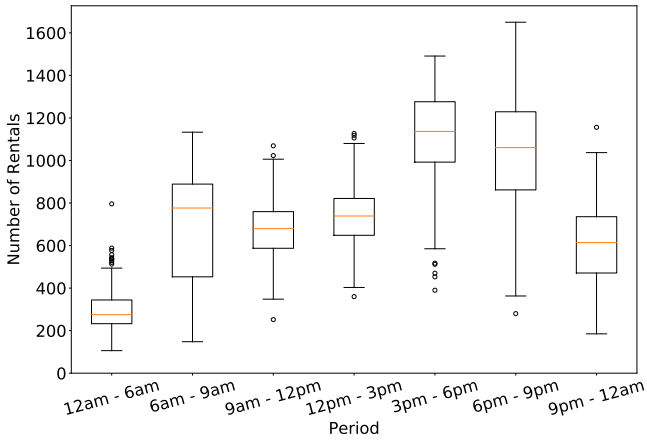
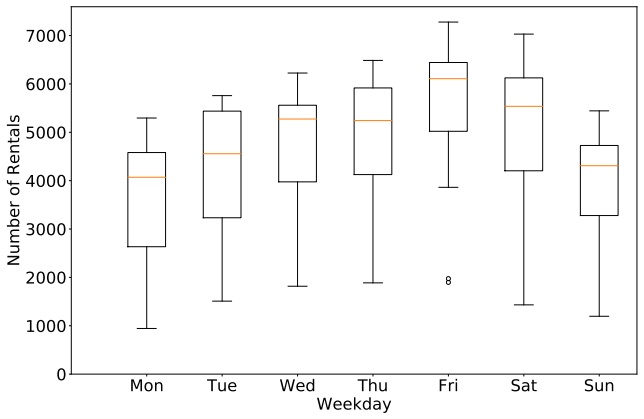


Figure 8.1: Time series of starting rentals in September 2017 aggregated per hour. The difference in the total number of time bins and the actual number of hours in the month are due to missing data (crawler failures).

We 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 we can spot some periodicity, there is a lot of variability that makes the prediction problem not straightforward. For our 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 (**schmoller2015empirical**). The rationale is to provide the FFCS company that actionable information on the demand for cars, e.g., to schedule car maintenance or



(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).

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

of the distribution, with outliers shown as points.[8] The series shows large variability, with peaks during early mornings (6am-9am) and afternoon (3pm-6pm and 6pm-9pm), 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.

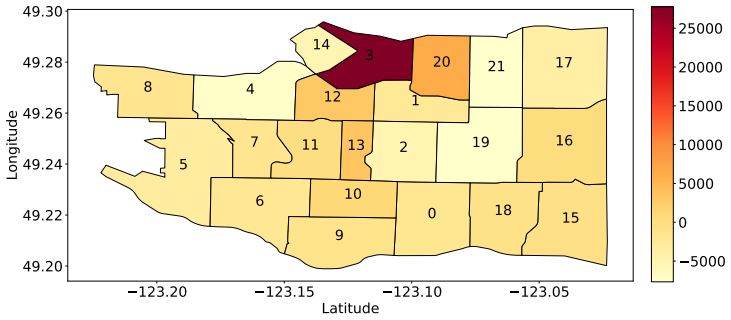
8.5.2 FFCS spatial characterization

We now take a closer look into 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), we here focus on particular examples to showcase the spatial variability in the demand for cars. We focus on the morning and afternoon peak time bins (6am-9am and 6pm-9pm). For each neighborhood we 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. We 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, we can 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.

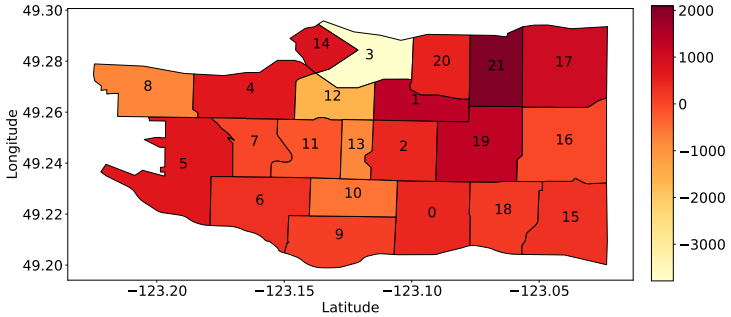
8.5.3 Socio-demographic and weather data characterisation

We 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, we 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.

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



(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)

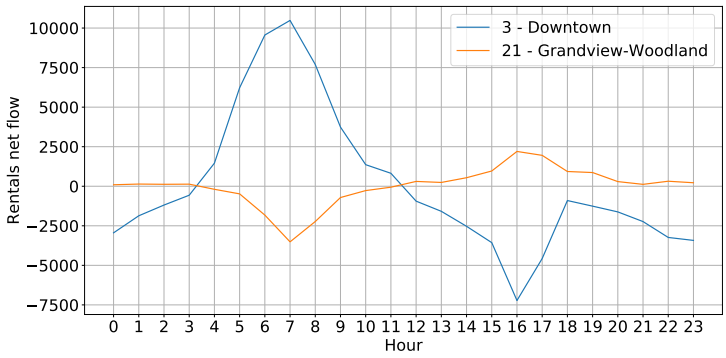


Figure 8.4: Total net flow in September 2017 for Downtown (neighborhood 3) and Grandview-Woodland (neighborhood 21) over different hours of the day

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, 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.

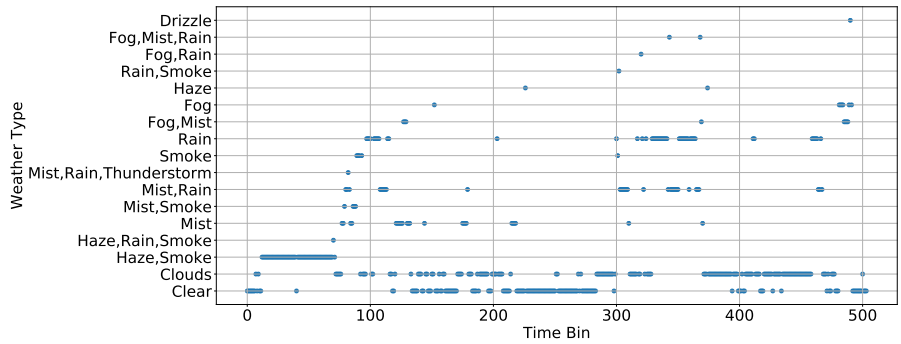


Figure 8.5: Time series of weather conditions per hour during September 2017. Each point in the plot represents an occurred weather type.

8.6 Temporal predictions of rentals

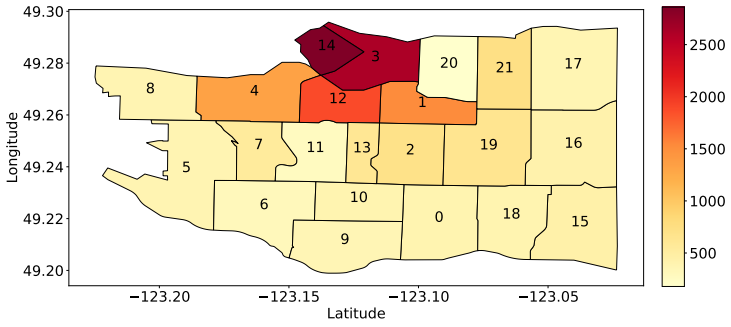
In this section, we describe our task of predicting the number of rentals in the whole city at a given time in the future. Eventually, the same methodology could be applied for each neighborhood. This prediction can exploit historical data, i.e., given the time series of rentals in the past, predict 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 our target variable, i.e., the number of rentals at time t. In the case of prediction with historical data, we predict

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

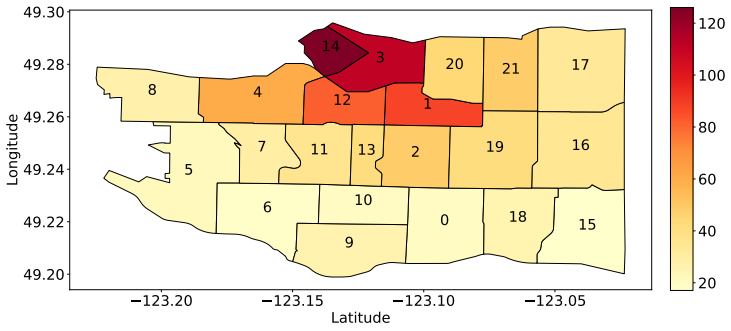
as a function $f()$ of the past $k+1$ data points of x itself. j is the horizon of the prediction.

If we also have other information, we can build a more generic model to consider the dependence to other variables. We want to predict

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



(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.

where $\{y_i\}$ are different variables – possibly other time series themselves (including x) – and g is the model that allows us 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.

Considering the time horizon of the prediction, we can 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 data at our disposal 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.

In this work, we consider discrete time, i.e., we split time into fixed size time intervals as defined in the aggregation step – see Section 8.5. We then build and train several machine learning models to tackle each aforementioned problem. Our 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

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

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

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

And the RMSE is defined as

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

8.6.1 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 (Jain, 1990), 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`[9] (Pedregosa

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

et al., 2011) and Keras[10]. Our code for the analysis is publicly available[11]. For details about each model, we refer the reader to (Bishop, 2006). In our implementations, we 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 our 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 our 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 our grid search to find the best parameters values to the range $[0, 3]$. The combination that gave us 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 our previous ARIMA model with an additional explicit daily seasonal component ($p = 7$ as the number of time bins in a day in our case).

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

SVR. In our 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

[10] https://keras.io/

[11] https://github.com/dougct/carsharing-prediction

most important features for the prediction problem. Here, we use 50 decision trees[12]. 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 Network model. LSTMs have a memory that helps capturing past trends in the data, which may favor our 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 our 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.

8.6.2 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 our 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. Our 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.

[12]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.

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.

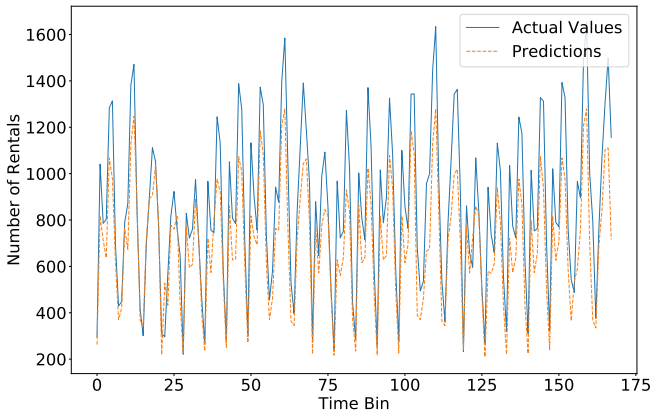


Figure 8.7: Long-term temporal prediction - Performance of the RFR model in one month of the validation set. The difference in the total number of time bins and the actual number of hours in the month are due to missing data (crawler failures).

8.6.3 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.

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 our 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 our 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)			Sliding Window (28	
Prediction Model	**MAPE [%]**	**σ(APE) [%]**	**RMSE**	**MAPE [%]**	**σ(APE) [%**
Baseline	20.12	16.64	195.53	20.12	16.6
ARIMA	36.01	35.87	306.80	36.52	36.6
SARIMA	17.60	20.01	160.42	18.02	21.7
Linear Regression	18.28	20.38	179.11	18.11	20.5
Support Vector Regression	12.22	15.62	128.72	12.87	18.5
Random Forests Regression	9.71	8.34	104.99	10.08	12.2
Neural Networks	10.52	12.93	128.84	10.52	12.7

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.

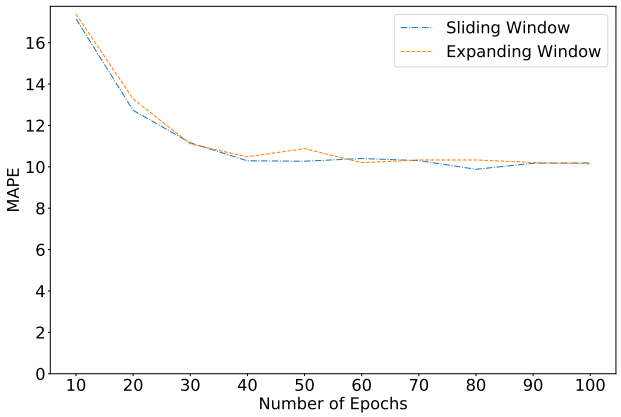


Figure 8.8: Effect of the number of epochs on the performance (MAPE) of the Neural Networks model.

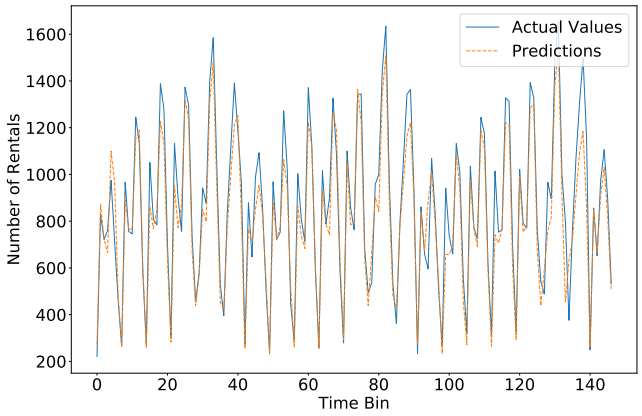


Figure 8.9: Short-term temporal prediction - Performance of the RFR model with expanding window in the validation set (24 days).

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 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.

8.6.4 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 our case, three months). In order to validate the effect of weather in this idealized situation, we assumed such perfect forecast and evaluated our 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. Our 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 our 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 our results. Indeed, our 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.

8.7 Spatial prediction of rentals with socio-demographic data

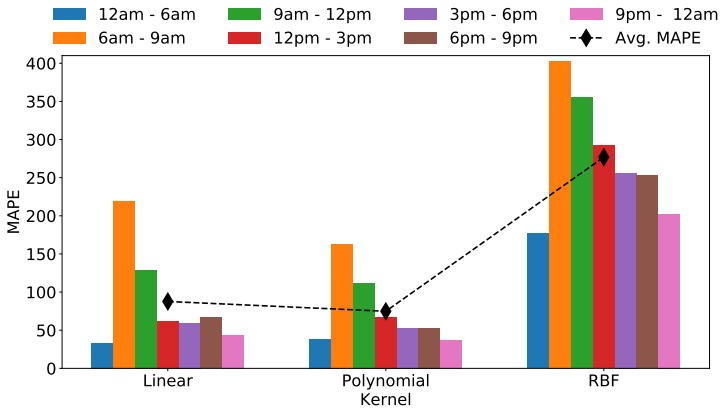
We now shift our attention to predict the demand of cars in a neighborhood without using past data as features. In other words, given only socio-demographic data in the neighborhoods, we try to predict the average number of expected rentals at each time bin, and at each neighborhood. This problem is 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 we have 22 neighborhoods which constitute our dataset for the training step, we could suffer from an overfitting problem. To minimize this potential effect, we follow a state-of-the-art approach, namely leave-one-out testing: given a target neighborhood, we consider information from all other neighborhoods for training the learning model, and consider the neighborhood that we left out for validation.

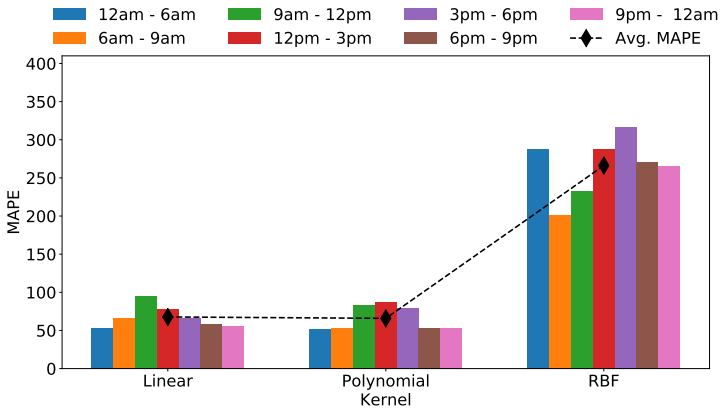
We manually select 83 socio-demographic features that we think might be related to human mobility. Here, we only apply the Support Vector Regression and Random Forest

Regression models, given that they were the best performing models (aside from neural networks) in the temporal prediction. We 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 (Bishop, 2006).

Considering hyperparameter tuning, for SVR, we 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, we try number of trees ranging from 10 to 100. We show the impact of hyperparameter tuning in the following.



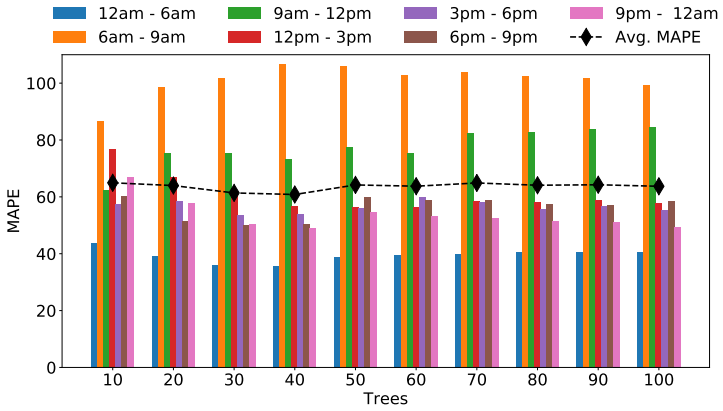
(a) SVR Model: Starting rentals



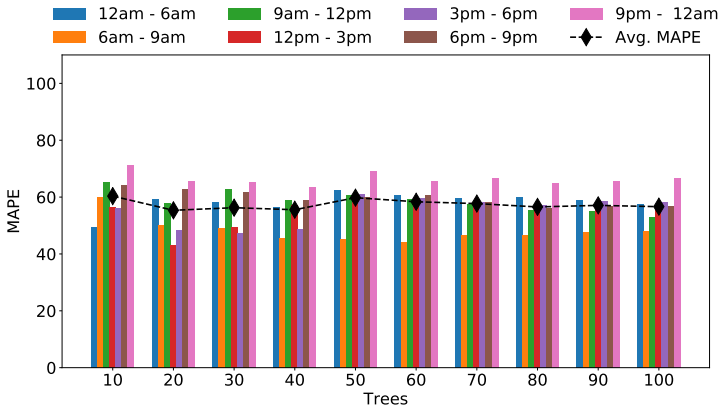
(b) SVR Model: Ending rentals

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

Figures 8.10a and 8.11b show the SVR prediction accuracy for the task of predicting



(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.

the number of starting and ending rentals, respectively. For each kernel type and for each time bin, we 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, we 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, we 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 we 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, we look into which features are the most important so to also perform feature selection and possibly improve the model.

8.7.1 Feature ranking and selection

As in the previous section, we 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 allows us to reduce the number of features to use in the model: we can focus only on the most important ones.

To evaluate the impact of the features on the model performance, we train once again the RFR with an increasing number of features, chosen according to the given rank. We 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 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.

We 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 8.1. 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.

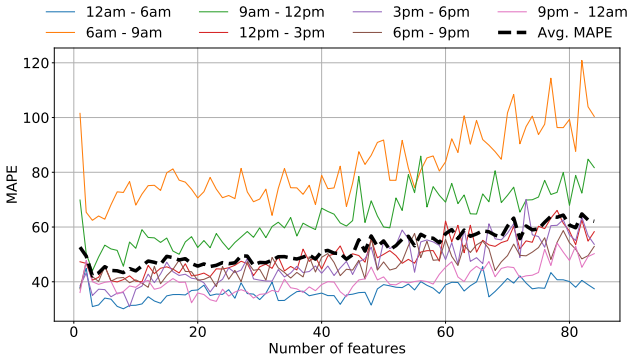
Finally, we explore the spatial prediction error, i.e., we look if there are neighborhoods that present significantly higher errors than others. Figure 8.13 depicts the heatmap of the MAPE per neighborhood, averaged over all time bins. The more the area is red the higher the average MAPE is. Each green 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. We can see that 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 our model assumes the operative area coincides with the total area of each neighborhood. Thus, our 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.

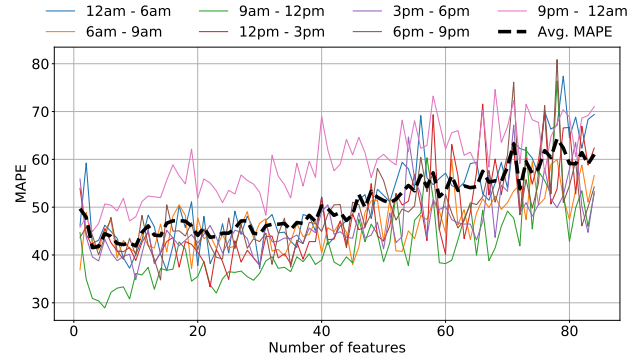
In general, the performance of the spatial predictions is lower when compared to the temporal predictions. This is expected given the nature of the problem, the limited amount of available data, and because the number of rentals varies widely within each neighborhood. However, we would like to emphasize 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.

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



(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

8.8 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

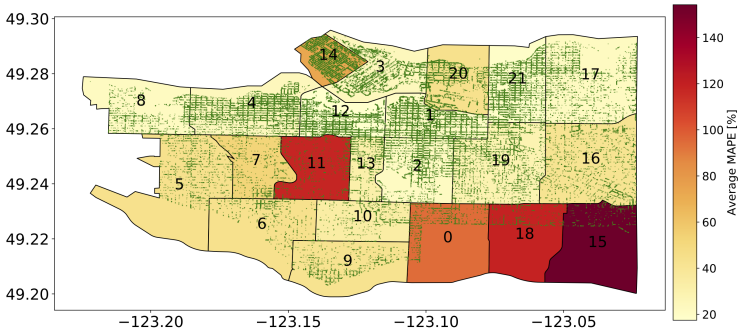


Figure 8.13: Spatial distribution - Heatmap of average MAPE per neighborhood. Rentals are shown on the map as green points

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 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.

Chapter 9

Relocation

In questo capitolo dimostro come le CS tengono comunque bilanciata la domanda e quindi non c'è bisogno di reloc (se riesco a pubblicare questo lavoro)

Chapter 10

Scalability

Ha senso mettere qui anche il lavoro con Alessandro e Michelangelo? non ho contribuito parecchio pero sembra aderire a concludere abbastanza bene la tesi

Chapter 11

Conclusions

Conclusion here

Bibliography

Alencar, Victor A., Felipe Rooke, Michele Cocca, Luca Vassio, Jussara Almeida, and Alex Borges Vieira (2019), "Characterizing client usage patterns and service demand for car-sharing systems," *Information Systems*, p. 101448, issn: 0306-4379, doi: https://doi.org/10.1016/j.is.2019.101448, http://www.sciencedirect.com/science/article/pii/S0306437919305009.

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

Bi, Ran, Jiajian Xiao, Dominik Pelzer, David Ciechanowicz, David Eckhoff, and Alois Knoll (2017), "A simulation-based heuristic for city-scale electric vehicle charging station placement," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pp. 1-7.

Bishop, Christopher M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg, isbn: 0387310738.

Boldrini, Chiara, Raffaele Bruno, and Marco Conti (2016), "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)*, pp. 572-577.

Brendel, Alfred, Sascha Lichtenberg, Ilja Nastjuk, and Lutz M Kolbe (2017), "Adapting Carsharing Vehicle Relocation Strategies for Shared Autonomous Electric Vehicle Services," in

Brendel, Alfred, Julian Tim Brennecke, Patryk Zapadka, and Lutz M Kolbe (n.d.), "A Decision Support System for Computation of Carsharing Pricing Areas and its Influence on Vehicle Distribution," in

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

Brockwell, Peter J and Richard A Davis (2016), *Introduction to time series and forecasting*, springer.

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

car2go API (n.d.), https://github.com/car2go/openAPI, Accessed: 10/03/2017.

Catalano, Mario, Barbara Lo Casto, and Marco Migliore (2008), "Car sharing demand estimation and urban transport demand modelling using stated preference techniques," *European Transport*, 40, pp. 33-50, issn: 1825-3997.

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

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

Ciari, F., N. Schuessler, and K. W. Axhausen (2013), "Estimation of carsharing demand using an activity-based microsimulation approach: model discussion and some results," *International Journal of Sustainable Transportation*, 7, 1, pp. 70-84.

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

Ciari, Francesco, Claude Weis, and Milos Balac (2016), "Evaluating the influence of carsharing stations' location on potential membership: a Swiss case study," *EURO Journal on Transportation and Logistics*, 5, 3, pp. 345-369.

Ciociola, A., M. Cocca, D. Giordano, M. Mellia, A. Morichetta, A. Putina, and F. Salutari (2017), "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)*, pp. 1-8, doi: 10.1109/UIC-ATC.2017.8397566.

Clark, Stephen (2003), "Traffic prediction using multivariate nonparametric regression," *Journal of transportation engineering*, 129, 2, pp. 161-168.

Cocca, M., D. Giordano, M. Mellia, and L. Vassio (2019), "Free Floating Electric Car Sharing: A Data Driven Approach for System Design," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1-13, doi: 10.1109/TITS.2019.2932809.

Cocca, Michele, Danilo Giordano, Marco Mellia, and Luca Vassio (2018), "A Data Driven Optimization of Charging Station Placement for EV Free Floating Car Sharing," in *21st IEEE International Conference on Intelligent Transportation Systems*.

— (2019), "Free floating electric car sharing design: Data driven optimisation," *Pervasive and Mobile Computing*, 55, pp. 59-75, issn: 1574-1192, doi: 10.1016/j.pmcj.2019.02.007, http://www.sciencedirect.com/science/article/pii/S1574119218305170.

Cocca, Michele, Danilo Giordano, Luca Vassio, and Marco Mellia (2018), "Free Floating Electric Car Sharing in Smart Cities: Data Driven System Dimensioning," in *4th IEEE International Conference on Smart Computing*.

Eisel, Matthias, Björn Hildebrandt, Lutz Kolbe, and Johannes Schmidt (2015), "Applying Demand Response Programs for Electric Vehicle Fleets," in *AMCIS*.

Falvo, Maria Carmen, Danilo Sbordone, Safak Bayram, and Michael Devetsikiotis (2014), "EV charging stations and modes: International standards," in *2014 Symposium on Power Electronics, Electrical Drives, Automation and Motion*, doi: 10.1109/SPEEDAM.2014.6872107.

Ferrero, F., G. Perboli, S. Musso, and A. G. A. Vesco (2016), "Business models and tariff simulation in car-sharing services."

Firnkorn, Jörg (2012), "Triangulation of two methods measuring the impacts of a free-floating carsharing system in Germany," *Transportation Research Part A: Policy and Practice*, 46, 10, pp. 1654-1672, issn: 0965-8564, doi: http://dx.doi.org/10.1016/j.tra.2012.08.003, http://www.sciencedirect.com/science/article/pii/S0965856412001334.

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

Firnkorn, Jörg and Martin Müller (2011), "What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm," *Ecological Economics*, 70, 8, pp. 1519-1528, issn: 0921-8009, doi: http://dx.doi.org/10.1016/j.ecolecon.2011.03.014, http://www.sciencedirect.com/science/article/pii/S0921800911001030.

— (2012), "Selling Mobility instead of Cars: New Business Strategies of Automakers and the Impact on Private Vehicle Holding," *Business Strategy and the Environment*, 21, 4, pp. 264-280, issn: 1099-0836, doi: 10.1002/bse.738.

Flath, Christoph, Jens Ilg, and Christof Weinhardt (2012), "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.*

Floudas, Christodoulos C. A. and P. M. Pardalos (2006), *Encyclopedia of Optimization*, Springer-Verlag, isbn: 0387336249.

Goldberg, David E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st, Addison-Wesley Longman Publishing, isbn: 0201157675.

Habibi, Shiva, Frances Sprei, Cristofer Englundn, Stefan Pettersson, Alex Voronov, Johan Wedlin, and Henrik Engdahl (2016), "Comparison of free-floating car sharing services in cities," in *European Council of Energy Efficient Economy Summer Study*, pp. 771-778.

Harms, Sylvia and Bernard Truffer (1998), "The emergence of a nation-wide car-sharing co-operative in Switzerland," *A case-study for the EC-supported rsearch project "Strategic Niche Management as a tool for transition to a sustainable transport system", EAWAG: Zürich.*

He, Suining and Kang G. Shin (2019), "Spatio-temporal Adaptive Pricing for Balancing Mobility-on-Demand Networks," *ACM Trans. Intell. Syst. Technol.*, 10, 4, 39:1-39:28, ISSN: 2157-6904, DOI: 10.1145/3331450, http://doi.acm.org/10.1145/3331450.

Herrera, Juan C, Daniel B Work, Ryan Herring, Xuegang Jeff Ban, Quinn Jacobson, and Alexandre M Bayen (2010), "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment," *Transportation Research Part C: Emerging Technologies*, 18, 4, pp. 568-583.

Herrmann, Sascha, Frederik Schulte, and Stefan Voß (2014), "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, pp. 151-162.

Hess, Andrea, Francesco Malandrino, Moritz Bastian Reinhardt, Claudio Casetti, Karin Anna Hummel, and Jose M. Barceló-Ordinas (2012), "Optimal Deployment of Charging Stations for Electric Vehicular Networks," in *Proceedings of the First Workshop on Urban Networking*, Nice, France, pp. 1-6, ISBN: 978-1-4503-1781-8.

Hulot, Pierre, Daniel Aloise, and Sanjay Dominik Jena (2018), "Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, ACM, London, United Kingdom, pp. 378-386, ISBN: 978-1-4503-5552-0, DOI: 10.1145/3219819.3219873, http://doi.acm.org/10.1145/3219819.3219873.

Jain, Raj (1990), *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, John Wiley & Sons.

Jiao, Zihao, Lun Ran, Jing Chen, Hui Meng, and Chong Li (2016), "Data-Driven Approach to Operation and Location Considering Range Anxiety of One-Way Electric Vehicles Sharing System," in *8th International Conference on Applied Energy*.

Jung, Malte F., David Sirkin, Turgut M. Gür, and Martin Steinert (2015), "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*.

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

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

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

Marc, Sonneberg, Kuehne Kathrin, and Michaelm Breitner (2015), "A Decision Support System for the Optimization of Electric Car Sharing Stations," in

Martin, Elliot and Susan Shaheen (2011), "The impact of carsharing on public transit and non-motorized travel: an exploration of North American carsharing survey data," *Energies*, 4, 11, pp. 2094-2114.

Millard-Ball, Adam (2005), *Car-sharing: Where and how it succeeds*, Transportation Research Board, vol. 108.

Okutani, Iwao and Yorgos J Stephanedes (1984), "Dynamic prediction of traffic volume through Kalman filtering theory," *Transportation Research Part B: Methodological*, 18, 1, pp. 1-11.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 12, pp. 2825-2830.

Phonrattanasak, P. and N. Leeprechanon (2014), "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)*.

Rao, Singiresu S. (2009), *Engineering Optimization: Theory and Practice: Fourth Edition*, English, John Wiley and Sons, isbn: 9780470183526.

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

Rooke, Felipe, Victor A. Alencar, Alex B. Vieira, Jussara M. Almeida, and Drago Idilio (2019), "Characterizing Usage Patterns and Service Demand of a Two-Way Car-Sharing System," in *Big Social Data and Urban Computing*, Springer Nature Switzerland AG, pp. 01-15.

Rooke, Felipe, Victor Aquiles, Alex Borges Vieira, Douglas do Couto Teixeira, Jussara M. Almeida, and Idilio Drago (2018), "Caracterização de Padrões de Demanda e Uso de um Sistema de Compartilhamento de Veículos de Duas Vias," *Workshop de Computação Urbana COURB-SBRC*, 2, 1/2018, issn: 2595-2706, http://portaldeconteudo.sbc.org.br/index.php/courb/article/view/2341.

Schmöller, Stefan and Klaus Bogenberger (2014), "Analyzing external factors on the spatial and temporal demand of car sharing systems," *Procedia-Social and Behavioral Sciences*, 111, pp. 8-17.

Schmöller, Stefan, Simone Weikl, Johannes Möller, and Klaus Bogenberger (2015), "Empirical analysis of free-floating carsharing usage: The Munich and Berlin case," *Transportation Research Part C: Emerging Technologies*, 56, pp. 34-51, issn: 0968-090X, doi: http://dx.doi.org/10.1016/j.trc.2015.03.008, http://www.sciencedirect.com/science/article/pii/S0968090X1500087X.

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

Shaheen, Susan A, Caroline Rodier, Gail Murray, Adam Cohen, and Elliot Martin (2010), *Carsharing and public parking policies: assessing benefits, costs, and best practices in North America*, tech. rep.

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

Smith, M. and J. Castellano (2015), *Costs Associated With Non-Residential Electric Vehicle Supply Equipment: Factors to consider in the implementation of electric vehicle charging stations*, tech. rep.

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

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

UMAP and electric car sharing simulator. Anonymized datasaset of 2 months of trips of car sharing users in the city of Turin (n.d.), https://github.com/michelelt/sim3.0.

Wagner, S., T. Brandt, and D. Neumann (2015), "Data Analytics in Free-Floating Carsharing: Evidence from the City of Berlin," in *2015 48th Hawaii International Conference on System Sciences*, pp. 897-907, doi: 10.1109/HICSS.2015.112.

Wagner, Sebastian, Christoph Willing, Tobias Brandt, and Dirk Neumann (2015), "Data Analytics for Location-Based Services: Enabling User-Based Relocation of Carsharing Vehicles," in *ICIS*.

Wang, Dong, Wei Cao, Jian Li, and Jieping Ye (2017), "DeepSD: supply-demand prediction for online car-hailing services using deep neural networks," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, IEEE, pp. 243-254.

Wang, X., D. MacKenzie, and Z. Cui (2017), *Complement or Competitior? Comparing car2go and Transit Travel Times, Prices, and Usage Patterns in Seattle*, tech. rep.

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

Wright, Stephen J. (2015), "Coordinate Descent Algorithms," *Math. Program.*, 151, 1 (June 2015), pp. 3-34, issn: 0025-5610.

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.