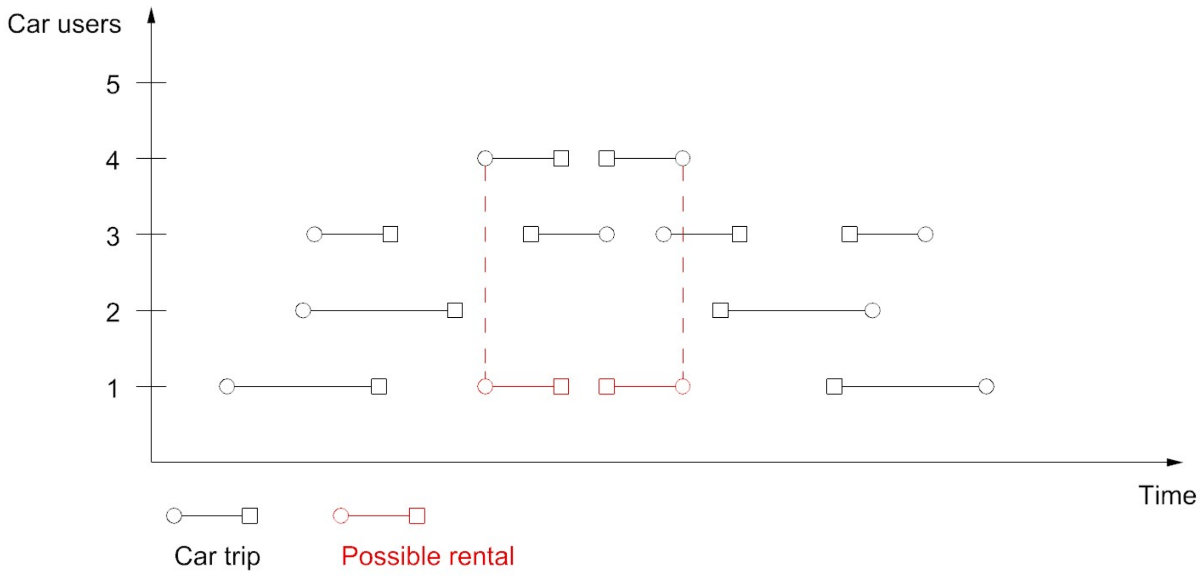


Impacts of Peer-to-Peer carsharing

Sarah Benz

Bachelor Thesis

August 2, 2022

Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Impacts of Peer-to-Peer carsharing

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	First name(s):
Benz	Sarah

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	Signature(s)
Zurich, 30.05.2022	S. Benz

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Acknowledgements

First of all, I would like to thank Prof. Dr. Kay W. Axhausen who supported me during the writing of my Bachelor Thesis and gave me constructive advice for the further development during the intermediate meetings. I would also like to thank Dr. Milos Balac for his support and insightful comments during the progress of working on my Bachelor Thesis.

Impacts of Peer-to-Peer carsharing

Sarah Benz
Student BSc in Civil Engineering
ETH Zürich
CH-8093 Zurich
sarabenz@student.ethz.ch

August 2, 2022

Abstract

At a time where climate change is a prevalent issue and many countries along Switzerland commit to reduce their CO_2 emissions, it is important to know the source of these emissions. In Switzerland, 40% of all CO_2 emissions are caused by traffic. Hereof, 72% are caused only by the driving of private vehicle. Combined with the fact that cars remain parked for 90% of the day, this provokes some reflection. The Peer-to-Peer (P2P) carsharing model starts precisely there and offers an approach to increase the efficiency of those cars and provides an alternative to car ownership. By sharing already privately owned cars, the P2P model would allow a reduction of the total cars needed for the car trip demand. This thesis, therefore, aims to estimate the number of cars needed to satisfy the current car trip demand in the greater area of Zurich, Switzerland. For this a heuristic approach was chosen. The results show that 40% of all currently available cars in the analysed area could satisfy the current car trip demand in the greater area of Zurich. Furthermore, the greatest potential was found to be during working hours (8 AM - 4 PM). Additionally, the influence of the walking distance on the share of cars needed decreases with increasing sample share. Although the availability of a cost-efficient alternative to car ownership might result in induced car usage, the implementation of a P2P carsharing model increases the efficiency of the used cars, extends mobility for people with no means of buying a car, and can even create an additional income for the vehicle owners.

Keywords

Carsharing, Peer-to-Peer, P2P, Transportation, Zurich

Suggested Citation

Benz, S. (2022) Impacts of Peer-to-Peer carsharing, Bachelor Thesis, Institute for Transport Planning and Systems (IVT), ETH Zurich, Zurich.

Contents

List of Figures

List of Tables

1 Introduction

In a world where climate change is a prevalent issue, it seems reasonable to think about how we can use our already owned resources to their fullest. In Europe, transport emissions make up 30% of the total CO_2 emissions with 72% being caused by cars (European Parliament, 2022). In Switzerland, traffic is responsible for 40% of the total emissions, not including international air traffic (Bundesamt für Statistik, 2022). Hereof, 72% alone are caused by the driving of private vehicles. The thought that those vehicles sit idle 90% of the time (Hampshire and Gaites, 2011) gives food for thought on how these resources can be utilized more efficiently.

One solution offers the Peer-to-Peer (P2P) carsharing model, where privately owned vehicles are rented out by their owners for other drivers for a short time interval of a few hours up to days (Shaheen and Cohen, 2020). An operator provides the environment, a platform, where owners can put up their vehicle and set a price for borrowing their car, and renters can book a vehicle fitting their needs. In addition, today most P2P operators do also provide a full coverage insurance, to ensure a risk free rental for all involved parties (2EM, 2022b; GoMore, 2022). In turn for providing the marketplace for the renters and owners, the operator either claims an initial 'participation fee', an annual or monthly fee, or they keep a part of the rent as a commission (usually around 20%) (Münzel *et al.*, 2019).

The P2P model seeks to make more efficient use of the already existing car resources by making them available to a greater public. Furthermore, at a time where people want to consume and access products and services at exactly the time, place and in the quantity and quality they desire (e.g. leasing) (Wilhelms *et al.*, 2017), the P2P carsharing model could make use of this demand and fill this market gap. In order to estimate the possible scope of the Peer-to-Peer model, the aim of this thesis is to investigate the potential of the Peer-to-Peer carsharing in the greater area of Zurich, Switzerland, by using a heuristic approach to estimate how many vehicles are needed to satisfy the current car trip demand.

The following chapter presents the history of carsharing and the concept of the Peer-to-Peer model, P2P operators in Switzerland are listed and different carsharing models compared. The chapter also touches upon the factors influencing P2P participation and its market potential. Consecutively, the methodology for calculating the minimum number of needed cars to satisfy the trip demand is explained, and the results are displayed. In Section 5 these results are discussed and limitations and recommendations for further research are highlighted. In the concluding chapter the discussion is wrapped up and the findings summarized.

2 Literature Review

2.1 History of carsharing (Focus on Switzerland)

Carsharing is a phenomenon that started to appear around 1950. One of the first carsharing models was the SEFAGE – Selbstfahrgemeinschaft in Zurich, Switzerland. In Switzerland, since the 18th century the high property prices led people, who couldn't afford them, to live together in 'Wohngenossenschaften' - resident's cooperatives. While the need for a car was there, in 1948 the hurdle for each resident to buy their own car made a dozen residents of a cooperative come up with the idea of owning a car as a cooperative for each resident to use – the idea of carsharing (Harms and Truffer, 1998; CERTU - CETE de Lyon, 2011).

The idea did not vanish and later in 1987 eight people from Stans founded the AutoTeilet Genossenschaft (ATG) while 17 people started the ShareCom in Zurich Seebach. Not long after their foundings the two signed a cooperative contract allowing the users access to both car fleets. As the demand continued to increase, an electronic reservation system was put in place in order to offer its members cheap and flexible car mobility without the need of buying a car (Harms and Truffer, 1998; CERTU - CETE de Lyon, 2011).

Finally, in 1997 the ATG and the ShareCom officially merged and established the Mobility carsharing (CERTU - CETE de Lyon, 2011) with a starting fleet of 760 vehicles for its 17,400 members. In 2016 Mobility carsharing counted a total of over 131,000 customers served by 2,950 cars which are stationed at 1,500 locations (Mobility Carsharing, 2022).

The fast growth of carsharing was not only observed in Switzerland, but in other countries as well. In 2007 carsharing took hold in over 600 cities worldwide with an estimated vehicle fleet of 11,700 (60% in Europe) (van der Linden, 2016). Although there are places where carsharing was not successful, such as Belgium (van der Linden, 2016), carsharing attracts new members every year and some operators achieve utilization rates of their fleet of 40% (Hampshire and Gaites, 2011). This means that the average vehicle in the operator's fleet is driven 9h out of 24h by a paying customer.

When we are talking about carsharing in general, most often 'traditional carsharing' is meant. Traditional carsharing describes an environment in which members can get access to a vehicle for short-term daily use. Traditional carsharing is not intended for longterm use of a car, but acts as a supplement for public transit. Those vehicles are owned or leased by a carsharing operator and are distributed throughout the network. They are either station-based, where the members have to park the rented vehicle at one of the operator's reserved parking spaces, or free-floating, where the renting person is only restricted to park the car within a defined area. When the members make use of such a vehicle, they are charged based on time and often per mile too. During the rise of traditional carsharing, another model of carsharing emerged around 2010 – the Peer-to-Peer (P2P) carsharing (Münzel *et al.*, 2018; Shaheen *et al.*, 2012). While the traditional carsharing appeared as a means to reduce expenses, the reasons for P2P are of environmental nature (Dill *et al.*, 2019; Münzel *et al.*, 2018). Making an underused available resource accessible to a larger group of people, the idea is to stop people from buying their own car but nonetheless providing them the comfort of having a car at hand when needed (Wilhelms *et al.*, 2017).

2.2 Concept of Peer-to-Peer carsharing

Peer-to-Peer (P2P) carsharing is a carsharing approach that has been around since 2010 (Münzel *et al.*, 2018). While traditional carsharing services operate their own fleet of vehicles, which they maintain and expand, P2P carsharing is based on private car owners making their vehicles available on a platform and renting it out for up to a few hours or days (Shaheen *et al.*, 2012). The idea behind P2P carsharing is to provide the users a car conforming to their wishes when they are in need of a vehicle. As some people only occasionally require a car, this service has the advantage that the users do not have to go to the length of buying and maintaining their own car. The result of P2P carsharing is not the reduction of the cars on the roads, but more the decrease of the number of cars sitting idle in parking (Hampshire and Gaites, 2011).

The environment for P2P is provided by third-party operators (nonprofit or for-profit), who maintain the platform where the renters and owners can rent or lend their car. These operators act as a middleman and ensure a positive experience on both sides, by making sure that both the car owners and drivers take good care of the vehicles. The P2P service operator verifies the insurance of the two sides and can act as a 'mediator' in disputes over e.g., car damage (Barbour *et al.*, 2020; Shaheen *et al.*, 2018).

2.3 P2P operators in Switzerland

From 2010 on, Peer-to-Peer started to be a greater phenomenon. Already as of May 2012 there were 33 P2P operators worldwide. 17 existed in North America, of which 10 were active or in pilot phase, three planned, and four already defunct (Shaheen *et al.*, 2012). In Switzerland two operators had found its marketspace, namely 2EM and Sharoo (2EM, 2022b; Mobility Carsharing, 2022).

This section provides an overview of the situation of P2P providers in Switzerland. Two active operators (2EM and GoMore) are presented, although GoMore only launched its service in October 2021. The third operator, Sharoo, discontinued its services after 6 years in May 2020 due to insufficient demand. It, therefore, also offers an insight into the limitations or hurdles P2P carsharing operators face in Switzerland.

2.3.1 2EM

2EM is one of the companies that provide an environment for private carsharing in Switzerland (2EM, 2022b). The founder Youness Felouati came up with the idea, as he lent his neighbor his car, as the neighbour's own car was not big enough for his errands at IKEA. His goal is to provide an alternative means of transportation convincing the users in four aspects: economic efficiency, convenience, environmental friendliness and social interaction. The service provided should allow a cost-efficient rent of a (for the situation) suitable vehicle of the same comfort, but without the actual need of buying and owning a car. One of the factors that allow 2EM to provide a cost-efficient experience and a free user and car registration is its non-profit model. Only if a rental takes place does 2EM secure a 22% commission of the rental price for covering the administrative costs. Although the platform provides suggestions for the rental price being based on , e.g., the car brand, age and fuel, the price is set by the vehicle owner himself. During their growth 2EM did not stop improving their service. While in earlier stages the renters and vehicle owners had to be insured themselves to take part, in 2017, five years after its establishment, 2EM managed to negotiate a contract with an insurance company to offer the possibility of a full coverage insurance if the vehicle owner does not have one. For this, a separate contract is signed between the owner and the renter, and the price is passed on to the renting member. Additionally, the renter is charged a deposit of 300 CHF until the end of the contract acting as an assurance for the vehicle owner. Furthermore, digitalization and new technology brought the chance to further facilitate the renting by installing a key box and eliminating the need for car renter and owner to meet in order to hand over the car key (2EM, 2022b). In ten years 2EM grew to be the largest P2P carsharing service in Switzerland with more than 35,000 members and over 2,200 registered vehicles. In comparison with Mobility - the biggest traditional carsharing company in Switzerland – 2EM has a smaller member community than Mobility with over 131,000, however the ratio of vehicle per member is greater by a factor of three (Mobility: 0.02, 2EM: 0.06) (2EM, 2022b; Mobility Carsharing, 2022).

2.3.2 GoMore

With 2.7 million members in Denmark, Sweden, Finland and Spain GoMore is striving since October 2021 to expand its services to Switzerland. With the goal to reduce the number of cars, particularly in cities, GoMore seeks to reduce CO_2 emissions and to use the existing resources more efficiently (GoMore, 2022). After only 4 months GoMore already records 500 registered cars with the most rentals taking place in Zurich (Baloise, 2022). With the insurance company Baloise as a partner, GoMore is able to provide their users full coverage insurance. While 2EM charges the owners 190 CHF for a keybox, GoMore bears the costs for the keybox. However, as GoMore predicts up to five times more rentals with a keyless access, they charge the owner with a keyless box 25 CHF per month, regardless of how many rentals took place that month (Bollinger, 2022; GoMore, 2022). Nevertheless, similar to 2EM, a GoMore registration is free of charge and they only keep a commission of 25% if a vehicle rental takes place. Just like 2EM, GoMore allows its car owners to set the rental price themselves (GoMore, 2022).

2.3.3 Sharoo

A third operator which provided a P2P carsharing service from May 2014 to May 2020 is Sharoo (Mobiliar, 2022; Rideable, 2022; 2EM, 2022a). Sharoo was established by Migros, Mobiliar and Mobility and started its service in Zurich, Berne and Lucerne. With the Sharoo-box installed on the car's dashboard, the bluetooth lock via the Sharoo app allowed the vehicle renters to access the car without the car owner having to be on site for the handover of keys (Mobiliar, 2022).

Although expectations were high, Sharoo missed their target by far. Their goal of having 10,000 registered cars by 2018 was too optimistic and in reality only 1,800 cars were available on the platform (Tagesanzeiger, 2022). Journalists saw several reasons for Sharoo's failure: monthly fees for vehicle owners, high prices, no possibility to handover car keys personally and an expensive installation of the Sharoo-box right at the beginning of the registration (Mobiliar, 2022; Tagesanzeiger, 2022; Rideable, 2022).

2.4 Differences between P2P and traditional carsharing services

Comparing the different carsharing models gives an insight into the advantages and disadvantages of individual models and how successful P2P is in comparison with other carsharing models.

2.4.1 Fleet size

In their studies Münzel *et al.* (2018) analysed 51 cooperative models, 43 B2C (business-to-consumer) roundtrip models, 4 B2C one-way models and 3 P2P carsharing models. The first great difference they analysed was the fleet size. While cooperatives in small towns count only few cars in their fleet, B2C roundtrip operators in larger cities have up to a few hundred. One-way B2C operators have over a thousand vehicles in their fleet and P2P carsharing have the largest fleets with up to multiple thousands. However, a large fleet size gives no information about how frequently the cars are rented out. As cooperatives and B2C operators offer their service only at places where the demand reaches a break even, P2P organizations are not bound by this (Münzel *et al.*, 2018; van der Linden, 2016). Nonetheless, the absolute fleet size gives a first hint at how successful the operators are. To see how dominant the models are in the potential market size, however, a vehicles per capita variable gives more insight. Interestingly, the operators show no substantial differences in vehicles per capita, meaning they are equally competitive in the area they operate in.

2.4.2 Partnerships

A second difference proved to be the number of partnerships with the operators. P2P providers showed few to no partnerships and the service was not provided by an incumbent, but more often by startups. The research explains this observation by the radically new 'disruptive' behaviour of the P2P model to the already existing carsharing market. In contrast to traditional carsharing, the P2P model does not require a new fleet of its own, but relies on private vehicle owners to provide their car to others (Münzel *et al.*, 2018; van der Linden, 2016; Hampshire and Gaites, 2011). This eases the financial burden on the startups, leading to the greatest expense being the marketplace they provide and the key card locks for the vehicles (Münzel *et al.*, 2018).

Similarly, Münzel *et al.* (2018) found that roundtrip B2C organizations were mostly established by startups too (74%). Unlike P2P operators, the other roundtrip types form partnerships with public transport organizations and city-related partners. Motives for the establishment of a roundtrip B2C service were proved to be mainly of environmental nature. A different case is the one-way B2C model. 75% of all one-way operators were incumbent, meaning they started in the carsharing business in addition to their main business model (e.g. car manufacturers). In contrast to roundtrip B2C models they do not primarily have environmental aspects in mind, but they seek to promote their main business. Furthermore, they show the most extensive network with lots of partnerships, such as public transport services.

2.4.3 Locations

Looking at the locations at which the carsharing organizations operate, it can be said that they service quite distinct areas. As already mentioned, cooperatives and B2C organizations are bound to locations, where the demand reaches a threshold for their business to be profitable (Münzel *et al.*, 2018; Hampshire and Gaites, 2011). This restricts their operating area to cities or larger towns, where they can be sure to attract customers to pay off their large fixed costs of operating their fleet. On the other hand, P2P type organizations are not restricted to city centers or denser areas. Due to their fleet being provided and maintained by the vehicle's owners, the marginal costs of supplying the cars is eliminated. This allows them to offer their services even in areas with little to no demand. The fact that P2P carsharing is essentially 'agnostic' has the advantage that P2P can be a means to increase the accessibility and flexibility of people who do not have sufficient public transport connections (Münzel *et al.*, 2018; van der Linden, 2016; Hampshire and Gaites, 2011; Shaheen *et al.*, 2012).

2.4.4 Price Structures

The last point compares the distinct price structures of the carsharing models. All models except the P2P type typically request a registration fee (cooperatives: 66%, B2C roundtrip: 64%, B2C one-way: 100%). 78% of the cooperatives and 64% of the B2C (roundtrip) operators additionally charge a monthly fee (Münzel *et al.*, 2018). Furthermore, both B2C models (93% roundtrip and 100% one-way) do normally demand payment on a minute and hour basis. This is the same for the P2P model (GoMore, 2022; 2EM, 2022b; Münzel *et al.*, 2018). This is understandable, as customers' needs of a car vary strongly (Münzel *et al.*, 2018; van der Linden, 2016). A person only using the carsharing service once a week will look negatively upon a monthly fee, whereas a member using the service daily will see a lot of advantages in paying a set, monthly fee. For the operators a minute-based or hourly fee poses a close to optimal solution with an attractive pricing system for less frequent, as well as for frequent users.

For P2P carsharing, there are currently several pricing structures being used. Some operators let the vehicle owners decide their own price, while others don't let the owners set the price themselves. At the moment, there appears to be a trend where the operators set the price for renting (hourly, daily) (Benjaafar *et al.*, 2019). A compromise is for the operator to give suggestions about the price to the owners, but letting them decide freely, or that the operators reserve the option to adjust the price upwards (Barbour *et al.*, 2020).

2.4.5 Rental costs

Analysing the actual rental costs, Münzel *et al.* (2018) and Wilhelms *et al.* (2017) found that rental prices for P2P carsharing are generally lower than the B2C alternatives. This stems from the fact, that the owners renting out their cars in P2P do not expect to generate or depend on a high additional revenue, but to make a little extra income by sharing their not-in-use car.

2.4.6 Convenience

Besides having the advantage of a lower cost for renting P2P vehicles, convenience can play a part in choosing P2P carsharing over the traditional models too. Wilhelms *et al.*

(2017) and Ballús-Armet *et al.* (2014) point out that people who have a need for distinct features (children's seat, etc.) of a car will have a greater chance to find what they are looking for with cars from a P2P service. In addition to that, levels of convenience will be on par with the other carsharing models as smart locks are installed and the personal key exchange falls away (Münzel *et al.*, 2018). On the other hand, in the survey Ballús-Armet *et al.* (2014) conducted in Oakland and San Francisco asking on the people's willigness to rent out their car, 17% respectively 25% expressed their concern over the reduction in availability and convenience of the car, as there will be times the car is not immediately available for a spontaneous trip.

In the end, Münzel *et al.* (2018) draws two possible futures for P2P carsharing. First, Peer-to-Peer carsharing becomes a serious competitor to traditional carsharing in small and large cities, however cooperatives continue to stay true to their purpose of the ideological and environmental principle of joint ownership and share a position with P2P operators in more rural areas. Second, P2P will be overtaken by one-way models of carsharing as their convenience level surpasses that of private carsharing. However, as the current models of traditional carsharing refrain from expanding to areas with less dense populations (Hampshire and Gaites, 2011), P2P could still fill this market gap and continue to avoid being completely overtaken by traditional carsharing models.

2.5 Factors influencing participation

To get an idea on how successful the Peer-to-Peer carsharing model can become, we can look at several studies (including stated preference surveys) and collect information about (possible) members of (P2P) carsharing services. Of the countless factors contributing to the decision on joining a P2P service as a renter or as a vehicle owner, hereafter the following aspects will be discussed in more depth: Additional income, age, education, gender, household composition, liability and insurance, and trust issues, followed by concluding thoughts on this section.

2.5.1 Additional income

The most relevant factor for the participation is the additional income. Although one might assume people take part in P2P carsharing for environmental reasons, research shows that economical factors play a much more crucial role in attracting new members.

Flick and Henseling (2019) state that economical motives play the most important role for the participation in P2P carsharing. While referring to market demand studies conducted in the United States and Europe, Hampshire and Gaites (2011) specifiy that cost savings alone are a convincing reason for 3% to 25% of the car drivers to give up car ownership, and instead obtain membership in a carsharing service. They emphasize that an economic incentive must be present for both renters and car owners to participate, i.e., for P2P carsharing to function.

2.5.2 Age

In a stated preference study Barbour *et al.* (2020) discovered that people of age 40 or greater had a higher probability to be extremely unlikely to rent their personal vehicle to others. Hampshire and Gaites (2011) go more into detail on this subject and observed that 37.6% of carsharing participants were of age 20-30, while only 27.6% were in the 30-40-year old age group. These findings are supported by the van der Linden (2016) study. With a quantitative research method (negative binomial regression model) van der Linden (2016) found that a larger percentage of the age group 20-24 would result in less shared cars. On the other hand, a larger group of people aged 25-34 years resulted in a higher number of shared P2P cars. Combining these findings, we can assume that the a P2P participant is most likely 25-30 years in age.

2.5.3 Education

(P2P) carsharing participants have been found to have a higher education on average than the general population (Shaheen *et al.*, 2018; Barbour *et al.*, 2020; van der Linden, 2016; Flick and Henseling, 2019; Dill *et al.*, 2019). In a survey of North American carsharing participants 43% had a bachelor's degree and 43% reported some postgraduate degree or an advanced degree. Only 2% of the participants had less than college education. In van der Linden (2016)'s model configuration the level of education showed to be statistically significant for the number of shared cars. In the studied areas with a greater group of the working age population qualified at level 5/6 ISCED (International Standard Classification of Education), more shared cars could be detected.

2.5.4 Gender

Findings show that a male person is more likely to be part of a P2P carsharing model than a female (Barbour *et al.*, 2020; Flick and Henseling, 2019; Dill *et al.*, 2019). While researching the factors that influence people's willingness to share their personal vehicle, Barbour *et al.* (2020) observed that women were more likely to be extremely unlikely to rent their personal cars. They explain it with a higher reliance on personal vehicles by female users, i.e. a lack of other transportation options. Research has shown, that women do not have the same mobility needs, and do face other dangers compared to men while using transportation (Barbour *et al.*, 2020; Shaheen *et al.*, 2018).

2.5.5 Household composition

Interestingly, research has a controversial opinion on how the household composition influences carsharing membership. According to Barbour *et al.* (2020) (stated preference survey) one-person households have a high probability to be extremely unlikely to share their car and this factor was found to be statistically significant. Whereas van der Linden (2016) support that one-person households lead to more shared cars. The results of van der Linden (2016), where several European cities' P2P operators and members were analysed, showed that with a one percentage increase in one-person households, an increase of 3.3% in the number of shared cars is expected. While Barbour *et al.* (2020) does not go into detail about possible reasons, van der Linden (2016) explains the result with a reduced car dependency in one-person households. According to them, households with children (at least two-people households) are more car dependent as it is less practical to travel by public transport. Specific features needed in their car may keep them from participating in carsharing services. Millard-Ball *et al.* (2005) also observed a high percentage of one-person households partaking in carsharing services (36% of participants living alone). Their results indicate an average household size of 2.02 people, which supports the statement that smaller households are more likely to take up membership of a carsharing service. One attempt at explaining this phenomenon is the fact that one-person households are far more common in areas surrounding pods. Here, 'pods' refers to locations with one or more carsharing vehicles. As favourable public transport connections and the availability of other transport modes is key for carsharing to succeed (van der Linden, 2016; Hampshire and Gaites, 2011), this connection is reasonable.

2.5.6 Liability and insurance

Oftentimes, liability and insurance pose one of the major concerns people have when being asked if they would take part in a P2P carsharing service (Ballús-Armet *et al.*, 2014). In their survey, Ballús-Armet *et al.* (2014) asked people what issues they see with participating in a P2P service: (a) as a vehicle renter and (b) as a vehicle provider. For (a), over 27% responded they had liability issues. In case (b) the number was even higher with over 47%. These findings are supported by Shaheen *et al.* (2012) and Barbour *et al.* (2020). Ballús-Armet *et al.* (2014) explain this issue with the wording of personal vehicle insurance policies. These insurance policies are designed for personal use only, meaning the insurance is not liable when the car is being rented or leased to others. Additionally, in many states in the US (except for California, Oregon, Washington) car owners who share their vehicle for commercial use will get their insurance coverage cancelled. In California, Oregon and Washington, however, personal carsharing is classified as non-commercial use, protecting the insured car owner from possible damages and liabilities on behalf of the person renting their vehicle (Ballús-Armet *et al.*, 2014). If the states have a personal vehicle sharing legislation, insurance costs represent 20% to 25% of the P2P operators overall costs according to Shaheen *et al.* (2012). One solution is to collect operational data, which has been seen to decrease the insurance premiums by 30% in the first year. On the other hand, car owners in states without a personal vehicle sharing legislation risk premium spikes resulting from increased use. This means, that when no favourable (policy) environment for personal vehicle sharing is present, motivating people to take part in a P2P carsharing model will be difficult (Shaheen *et al.*, 2012; Barbour *et al.*, 2020; Ballús-Armet *et al.*, 2014; Shaheen and Cohen, 2020).

Notably, in Switzerland both P2P carsharing operators have found an insurance company as a partner, who is ready to provide a full coverage insurance as part of the membership of the P2P service (2EM, 2022b; GoMore, 2022). This hurdle, therefore, seems to be more prevalent in the US and in other countries (Shaheen *et al.*, 2012; van der Linden, 2016).

2.5.7 Trust issues

Linked to the concerns regarding liability and insurance, many respondents in Ballús-Armet *et al.* (2014) expressed a lack of trust in others in regard to their personal belongings. Shaheen *et al.* (2012) support this statement as 21 out of 29 respondents in their survey identified the trust issue as one of the top three barriers of P2P carsharing. Suggestions to increase the trust in the P2P system include user rating and feedback system, operator screening and selection, as well as social networking. In addition to that, operators incorporate vehicle owner control over who can rent their vehicle, and some organizations screen the vehicles for maintenance issues, age, fuel efficiency, and model specifications. Furthermore, in verifying the age, identity and driving record, trust issues can further be reduced.

2.5.8 Typical P2P members

Concluding this section, one can say the typical P2P member participates to earn an additional income. The member is most probable to be 25-30 years of age, of higher education, male, and lives in a two-person household. If a person decided against taking part in a P2P service, the issues are most likely a lack of trust, which may be not yet addressed by the operator, and a problem with the vehicle insurance and liability (either costs too high or no way of insuring).

2.6 Market potential of Peer-to-Peer carsharing

The advantage of P2P carsharing over carsharing with its own fleet of vehicles is, that the market is far less segmented. As P2P is not only profitable in urban areas (Münzel *et al.*, 2019), this way of sharing your own vehicle can appeal to a much broader demographic or group of potential participants. The reason for this is that with P2P the operators do not have to maintain a car fleet, so many expenses are eliminated. An example is the case with Zipcar in 2009, where of the total $137M operating expenses, $93M or 68% could be accounted for by costs associated with operating vehicles, which include: leases, depreciation, parking, fuel, insurance, gain or loss on disposal of vehicles, accidents, repairs and maintenance, and also employee-related costs. This implies that 50% or more of the operating costs of a carsharing service could be annihilated with the P2P model (Dill *et al.*, 2019; Shaheen *et al.*, 2012).

This means that the fees can be lower than for traditional carsharing services. As the vehicle owners see the income from P2P platforms as a means of additional earnings, unlike traditional carsharing operators having to pay wages, the prices are generally lower in P2P (Wilhelms *et al.*, 2017; Münzel *et al.*, 2018; Dill *et al.*, 2019). It is, therefore, more economically consistent with lower-density neighborhoods than traditional carsharing

(Hampshire and Gaites, 2011). Because P2P extends even to smaller towns, the potential for car accessibility is greater than with traditional carsharing (Hampshire and Gaites, 2011). According to Loose (2016), one carsharing car in a city center replaces up to 20 private vehicles and carsharing users reduced their car ownership by 62%. Another study (Chen and Kockelman, 2016) found out that people participating in a carsharing program are expected to reduce their average individual transportation energy use and GHG (greenhouse gas) emissions by 51% upon joining a carsharing organization. However, Münzel *et al.* (2018) also point to the possibility of P2P being completely overtaken and replaced by one-way models in the future, as this model offers users more freedom than roundtrip P2P services.

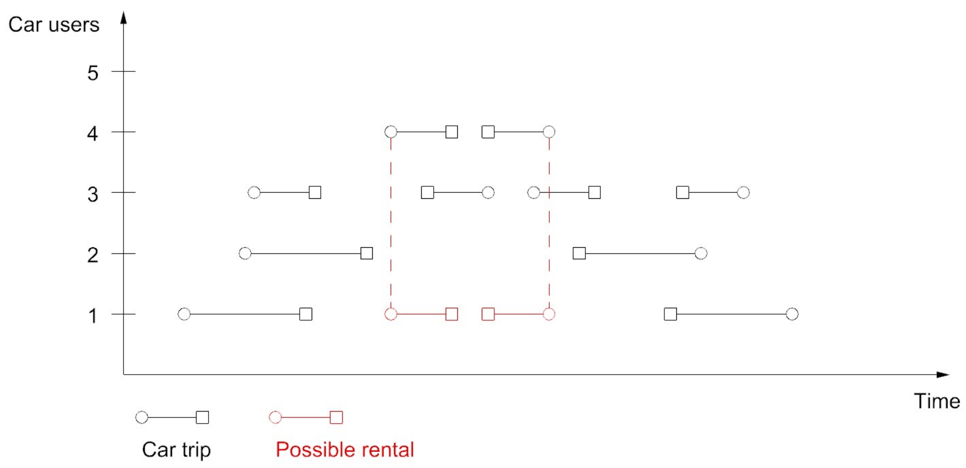
3 Methodology

This thesis uses the trips of agents representative of the studied region's population to estimate how many cars are needed to satisfy the current car trip demand, when a P2P carsharing model is implemented. The problem is solved in four different levels of detail: Approach 1, which is divided into 2 separate approaches, Approach 2 and Approach 3. For all calculations and the estimation of the cars needed to satisfy the car trip demand, Python Pandas was used on the EPFL Jupyter Notebooks (EPFL, 2022). Pandas is an open source data analysis and manipulation tool, which can be used in the Python programming language (Pandas, 2022). In a first step, the data set containing the agents' trips was cleaned and organised. As a second step, the data set was filtered to fit certain boundary conditions, e.g., that the transport mode was a car. Only then, additional data sets were created as a preparation for the implementation of the approaches. It is important to keep in mind that this thesis does, by no means, present an optimal solution. This study uses a heuristic approach and approaches the problem in the four different levels of detail which are explained in Section 3.6, Section 3.7 and Section 3.8. In the following subsection, the general idea of the implementation of the P2P model is given. Section 3.2 goes into detail about the origin and the cleaning of the original data set, while the subsections following thereafter will explain the implementation of the approaches in more depth. The exact programming code can be looked up in Appendix A.4.

3.1 General idea of the implementation

The general idea of the approaches is to find subtours which can fit into the trip schedule of a car that has already been used. A 'trip' is defined as the relocation from point A to point B. Furthermore, a subtour contains all the trips between leaving the location A (the agent's home) and returning to location A again. A subtour, therefore, includes at least two trips, one for leaving home and another trip to return home. In addition, an agent can have more than one subtour in one day, the first subtour being, e.g., leaving and returning for and from work, and the second subtour of the day can be to go and come back from grocery shopping. Fig. 1 gives an idea of how the P2P model will be implemented. We start with the subtour with the earliest departure time and add this car with its schedule to our needed cars. Next, we look at the second earliest subtour and see if this subtour fits into the first car's schedule. If that is the case, a rental has been found and the second person can take the rental car. In the situation of Fig. 1, however, car owner 2 travels at the same time as car owner 1. This means, car owner 2 has to take their own car too. Only with car 4 one can see a compatibility of the schedule with one of the earlier car owners. Mind that the P2P member has two options: Rent vehicle 1 or rent vehicle 2. In the implementation the P2P user will always take the first option that fulfills our conditions, in this case that would be vehicle 1. While Fig. 1 only shows the condition with the schedule compatibility, in the approaches we will look at two more conditions: The walking range to the car's location from the original trip start location cannot exceed either 500 m, 1,000 m, 1,500 m or 2,000 m, and the new travel time (with the detour by foot to the car's location) cannot be larger than 1.5 times the original travel time.

Figure 1: Illustration of the search for P2P rental options that are possible in terms of time



3.2 Data source

The data used for this thesis is taken from a 100% Zurich MATSim scenario. MATSim provides a framework to implement large-scale agent-based transport simulations and is open source (Horni *et al.*, 2016; Hörl *et al.*, 2018). On a micro-level scale it simulates all activities of the agents within the study area in a 30h-day. The reason for considering a 30h-day and not only a 24h-day lies with trips, that last for longer than until midnight. To account for these over-midnight-trips, a full day and 6h of the new day were considered. MATSim allows to create scenarios with a group of virtual people ('agents') that represent the travel behaviour of the population within the study area. Due to the socio-demographic, economic and behavioural aspects of the population being taken into account when creating the agents, MATSim offers a representative simulation of the travels undertaken in the study area.

The scenario that was used in this thesis has been provided by the Institute for Transport Planning and Systems at ETH. It is a cut-out of the national MATSim scenario that was created for Hörl (2020). The area was defined as the city of Zurich including a five kilometre buffer zone around the administrative boundary. The raw data analysed in this thesis is an equilibrium state of this MATSim simulation. During 40 iterations, all the agents 'tried' to optimise their travel time by, e.g., avoiding the simulated traffic or adjusting their mode. The results are a total of about 4.6M trips without any mode restriction.

These 4.6M trips are bundled in a csv file as the raw data. Each trip is assigned 23 variables (Fig. 2). As is visible in Fig. 2(a) and Fig. 2(b), there are variables where some values are either undefined or set to 'zero'. Although not all variables are needed for the investigation, some values such as the travel time ('trav_time') are expected to be different from zero. By dropping the trips with the same start and end location, all trips with zero travel time are eliminated (see Section 3.4).

Figure 2: Two example trips of the initial data set

```
person                                    201740002876460
trip_number                                             1
trip_id                                 201740002876460_1
dep_time                                         05:53:26
trav_time                                        00:47:13
wait_time                                        00:00:00
traveled_distance                                   24457
euclidean_distance                                  10391
main_mode                                             NaN
longest_distance_mode                                  pt
modes                        walk-pt-walk-pt-walk-pt-walk
start_activity_type                                  home
end_activity_type                                 outside
start_facility_id                  home201700971862030
start_link                                         245441
start_x                                         2682304.0
start_y                                         1255439.0
end_facility_id                                 outside_1
end_link                                           115634
end_x                                      2672923.791426
end_y                                      1250966.565433
first_pt_boarding_stop                                NaN
last_pt_egress_stop                                   NaN
Name: 0, dtype: object
```

(a) 1. trip of initial data set

```
person                          201740002877205
trip_number                                   4
trip_id                       201740002877205_4
dep_time                               12:07:24
trav_time                              00:00:00
wait_time                              00:00:00
traveled_distance                             0
euclidean_distance                            0
main_mode                                   NaN
longest_distance_mode                       NaN
modes                                       car
start_activity_type                        home
end_activity_type                          home
start_facility_id           home201702613637981
start_link                               699191
start_x                               2681107.0
start_y                               1246704.0
end_facility_id             home201702613637981
end_link                                 699191
end_x                                 2681107.0
end_y                                 1246704.0
first_pt_boarding_stop                      NaN
last_pt_egress_stop                         NaN
Name: 336, dtype: object
```

(b) 337th trip of initial data set

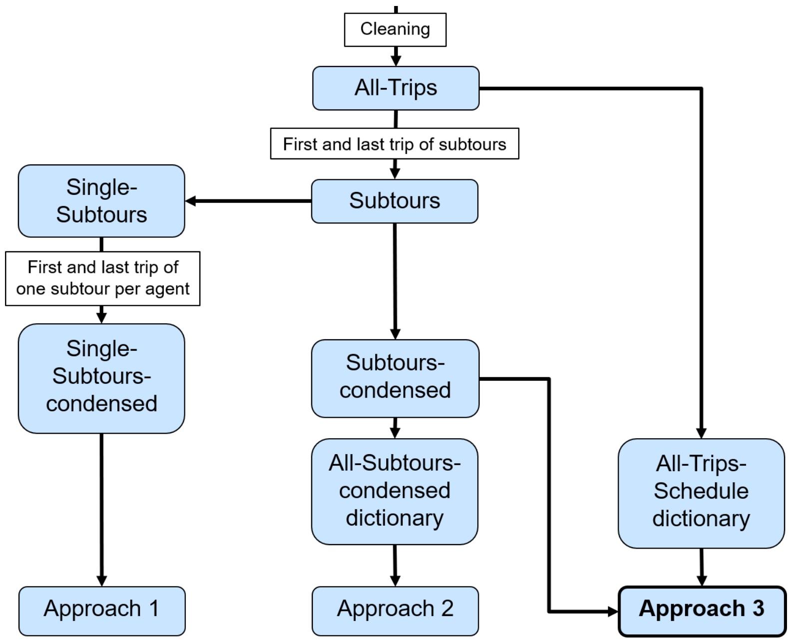
3.3 Overview of the different data sets

For comprehensibility reasons, Table 1 gives an overview of the data sets which will be created for the implementation of the different approaches and which characteristics the trips in the data set possess. In addition, Fig. 3 displays the connections between the different data sets, i.e. which data sets were used to create new ones and which data was taken. If nothing else is specified, all trips were adopted from the previous data set into the newly created one. Approach 3 is highlighted, as it is the most detailed and realistic approach, and, therefore, the most relevant. This is why we will concentrate on Approach 3 in the results section.

Table 1: Description of the used data sets

Data set	**Characteristics of the car trips**
All-Trips	Cleaned trips
Subtours	First and last trip of all subtours
Single-Subtours	First and last trip of first subtour of every agent
Single-Subtours-condensed	First and last trip of first subtour of every agent combined into one roundtrip
Subtours-condensed	First and last trip of all subtours combined into one roundtrip
All-Subtours-condensed dictionary	All subtours per agent collected in a dictionary
All-Trips-Schedule dictionary	All trips per agent collected in a dictionary

Figure 3: Overview of the connections between the data sets and their usage in the approaches



3.4 Cleaning of the data set (All-Trips)

Cleaning is the first and one of the most important steps in data analysis. An unstructured and incomplete data set will oftentimes yield useless or misleading results. The cleaning steps include primarily the removal of incomplete, faulty or duplicate trips. In this thesis, the process is divided into six steps, which will be discussed in the following subsections:

3.4.1 Changing of the time format

The initial data includes the variables 'dep_time' (departure time) and 'trav_time' (travel time), which are of the type 'string' ('xx:xx:xx'). To facilitate later calculations and the implementation of 'availability conditions', the format of these time values is changed to a 'numpy integer' (numpy.int64) representing the departure and travel time in seconds.

3.4.2 Dropping of non-car-mode trips

As we have seen in Fig. 2, the raw data includes various transport modes (car, walking, car-passenger, public transport, biking). To analyse the car tours we have to filter all trips by the 'modes' variable and drop the trips containing non-car transport modes.

3.4.3 Dropping of the zero distance trips

This data set does not only contain reasonable A to B travels, but also trips that have the same start and end location (cf. Fig. 2(b)). These trips falsify the results, therefore a condition is used to ensure different start and end locations in every trip: start_x + start_y - (end_x + end_y) > 1. The 'greater than 1' condition ensures that 'low-distance' trips are eliminated too (e.g. 0.5 m).

3.4.4 Sorting of the data set

To reduce computation time and increase the transparency of the data, unnecessary variables are dropped. The only variables kept are: ['person', 'dep_time', 'trav_time', 'start_x', 'start_y', 'end_x', 'end_y', 'trip_id']. To further facilitate the reading, the number of digits in the 'person' variable is reduced. In this thesis the data after these steps is called 'Filtered-trips'.

3.4.5 Consistency in the trip schedule

This step ensures that the agent's driving schedule is consistent. This means, that the agent cannot end his car trip at point B and start his next drive at point C. With the condition that the start-x-location of the following trip has to be equal to the end-x-location of the actual trip, we prevent having trips with 'location jumps' in the data set. Because P2P carsharing works with roundtrips, such jumps distort the results when the agents do not bring the rented vehicles back to their original location.

In the iteration, every agent's trips are checked on whether they are continuous in their locations, meaning the agents starts at location A, heads to locations B and C and returns to A again. When this is not the case, or only one trip of an agent is found, the subtour of this agent is incomplete and as a result the agent is dropped. Now the input data frame only has agents who take consistent trips.

3.4.6 Implementation of the roundtrip condition

Following on from the previous point, we drop all agents, who do not return to their initial location at the end of the day. This means checking the first and the last trip made by every agent and ensuring the agents return to their original location after they made all their trips. The reason for this is, that the P2P model only works, if the renting people return the car to original rental location where they picked up the car. As explained in Section 3.4.5, non-roundtrips would defeat the purpose of the P2P model. If the renters do not bring back the owner's vehicle, the owner does not have his own car available, which requires for him to look for another car. In the end, this leads to a free-floating model, which is not the topic of this thesis.

After the implementation of this condition, the data set is now consistent in the locations of the trips and shows no location jumps, and includes only trips where the owner returns to his initial location at the end of the day. For reference reasons this data set is called 'All-Trips'.

3.5 Preparation of the data sets for Approaches 1-3

There are two data sets that are used in almost all three approaches. The first one contains the first and last trip of every subtour. The second data set is a condensed version of the first data set, which contains the starting location and start and end time of the subtours.

3.5.1 Subtours

In order to create 'Subtours' with all the first and last trips of every subtour of all agents, the last trip of a subtour matching the first trip of the same subtour has to be identified. This is done by looking at each agent specifically and searching for a trip with the end location corresponding to the earliest departure trip's starting location. These two trips can then be added to 'Subtours'. If more the agent is doing more than one subtour in their day, the corresponding first and last trip of these subtours will also be added to 'Subtours'. At the end, 'Subtours' contains every first and last trip of all subtours (two trips per subtour) made by the agents.

3.5.2 Subtours-condensed

'Subtours-condensed' aims to combine the two separate trips of 'Subtours' into only one subtour. This subtour contains the agent, the starting/ending location as well as the start and end time of the subtour. After identifying the pairs of the first and last trip of a subtour in 'Subtours', 'Subtours-condensed' can be filled. While the start time of the subtour corresponds to the start time of the first trip, the end time of the subtour can be determined by adding the travel time to the start time of the last trip of the subtour. Due to the nature of a subtour, which is to return to its initial location, there is only a need for one location equal to both the initial and final location.

3.6 Approach 1

Approach 1 is the simplest approach and divided into Approach 1.1 and 1.2. In Approach 1 we look only at the first subtour of every agent. This means, only the trips until the agent returns to his starting location are taken into account. For full-time workers, this might take the whole day, for other agents who take several trips from and to their home, this would only be the first subtour (home - destination - home). Assuming every agent has their own car, in this approach we search for agents having a different schedule and starting their trips near each other to share a car. For this, some boundary conditions are set. First, the car is blocked from the moment the agent leaves his home until his return and, therefore, cannot be rented during his travels. Second, an agent wanting to rent a car is only willing to walk up to a defined distance to the car's location (see Table 2). Third, the person does not weigh the most optimal solution for them, but chooses the first one that is available and fulfills the boundary conditions.

Table 2: Variations in the walking range and walking time

Walking range	**Walking time** (v = 5 km/h)
500 m	360 sec.
1,000 m	720 sec.
1,500 m	1,080 sec.
2,000 m	1,440 sec.

3.6.1 Single-Subtours

The first preparation for approaches 1.1 and 2.2 is to generate a data set with only the trips of the first subtour of every agent, as the goal for these two approaches is to estimate the needed vehicles when every agent makes one subtour in a day. By identifying the earliest and second earliest trip of every agent in 'Subtours', the first and last trip of every first subtour of every agent is found and can be appended to 'Single-Subtours'.

3.6.2 Single-Subtours-condensed

To evade having to keep track of two trips per subtour for Approach 1.1 and 1.2, the information in 'Single-Subtours' is reduced to only one trip per subtour. The steps are the same as with 'Subtours-condensed': Locating the first and last trip of the subtour and combining this information in a new data set 'Single-Subtours-condensed'. The data includes: agent, starting time, ending time (= starting time + travel time), and location. After 'Single-Subtours-condensed' is filled with the information of the first subtour of every agent, Approach 1.1 is executed to find the number of cars needed to satisfy the agents' first subtour demand.

3.6.3 Implementation of Approach 1.1

Approach 1.1 looks at the first subtour of every agent and searches for possible P2P rentals in a radius of either 500 m, 1,000 m, 1,500 m, or 2,000 m. For a vehicle to be available, it has to be stationed at home and the rental's schedule cannot overlap with the car owner's schedule. This means that while the owner is away from home (starting location of the very first trip), the car cannot be rented. The idea for this approach is to keep track of the 'needed-cars', which includes all the vehicles which are needed to satisfy the car trip demand. While 'needed-cars' is empty at the beginning, the agents' car is added if there is no rental option available. At the end, the length of the 'needed-cars' dictionary shows the minimum number of cars for this approach.

The general outline of Approach 1.1 includes two loops, one inside the other. The outer loop indicates the subtour, for which a rental should be found, and the inner loop iterates over all possible rental cars. Should the inner loop not yield a successful rental, the agent takes their own car to make the subtour and their car is added to the possible rental cars in the inner loop.

To see if there is a possible rental, at first, the walking distance from the agent's location to the car's location is calculated. This is done by calculating the linear distance between the locations and multiplying it by a detour factor of 1.3 to account for turns and detours due to, e.g., buildings. By assuming an average walking velocity of 5 km/h the walking time to and from the car (at the beginning and the end of the subtour) can be calculated. Adding the walking time to the original ending time of the subtour results in the travel time if the car was rented.

In a final step, the time compatibility is checked and the condition that the walking distance cannot exceed a certain range (500 m, 1,000 m, 1,500 m, 2,000 m) is verified. Given that both conditions are fulfilled, the car is rented. In the end, if no suitable rental car is found, the inner loop terminates without a renting match, meaning the owner's car is added to the rental cars. The running of this code results in a filled 'needed-cars', where only the first subtour of every agent is considered, the car can only be rented at home, and double bookings are possible.

3.6.4 Implementation of Approach 1.2

Approach 1.2 is very similar to Approach 1.1. The only difference is that the rental is now implemented in the vehicle's schedule. In other words, if a rental takes place, the car is blocked during the rental time and no other agent can rent this car during this time. Although the conceptual difference is small with only the rental schedule which is updated, the implementation now requires to keep track of all (not just one) subtours a car makes.

Same as Approach 1.1, Approach 1.2 is structured into an outer and an inner loop, where the outer loop indicates the subtour, for which a rental has to be found, and the inner loop, where the available rental cars are checked for the rental. Different in this approach is, that double bookings should not occur and, therefore, the car schedule has to be kept track of. This is implemented, by adding an iteration over the schedule of the considered 'needed-car'. After calculating the additional walking time for the rental (see Section 3.6.3) the time compatibility of the rental car with the subtour can be checked. If the subtour, for which a rental is to be found, isn't overlapping with any of the subtours of the schedule of the 'needed-car', and the walking distance is in the expected range (500 m, 1,000 m, 1,500 m, 2,000 m), the rental is successful. Then, the schedule of the 'needed-car' is updated to also contain the actual subtour. The car is now blocked during this subtour's rental time too. In the case no rental is found, the two inner loops terminate and the owner's car is added to the 'needed-cars'.

Therefore, Approach 1.2 returns a filled 'needed-cars' data set of which the length indicates the number of cars needed to fulfill the car trip demand (one subtour per agent).

3.7 Approach 2

Approach 2 is similar to Approach 1, as it only looks at roundtrips, meaning the car can only be rented at home, inbetween the subtours of the car's owner. However, Approach 2 implements not only the first subtour of every agent, but all subtours. This requires to keep track of the different subtours and find gaps in the schedule, where a rental can take place (like in Section 3.6.4). For this purpose, the data set 'All-Subtours-condensed' based on 'Subtours-condensed' is created.

3.7.1 All-Subtours-condensed (dictionary-type)

The 'All-Subtours-condensed' data set is a dictionary that is based on the data in 'Subtours-condensed'. The advantage of the dictionary in Approach 2 is, that no additional data sets (in this case 'Subtours-condensed' is of type data frame) have to be created when wanting to iterate over a car's schedule. This is important, as the agents can do more than one subtour in a day and the available time slots for the rental have to be identified. Therefore, all subtours of every car owner are identified and added to 'All-Subtours-condensed' with the car owner as the key, and the car's schedule as the value.

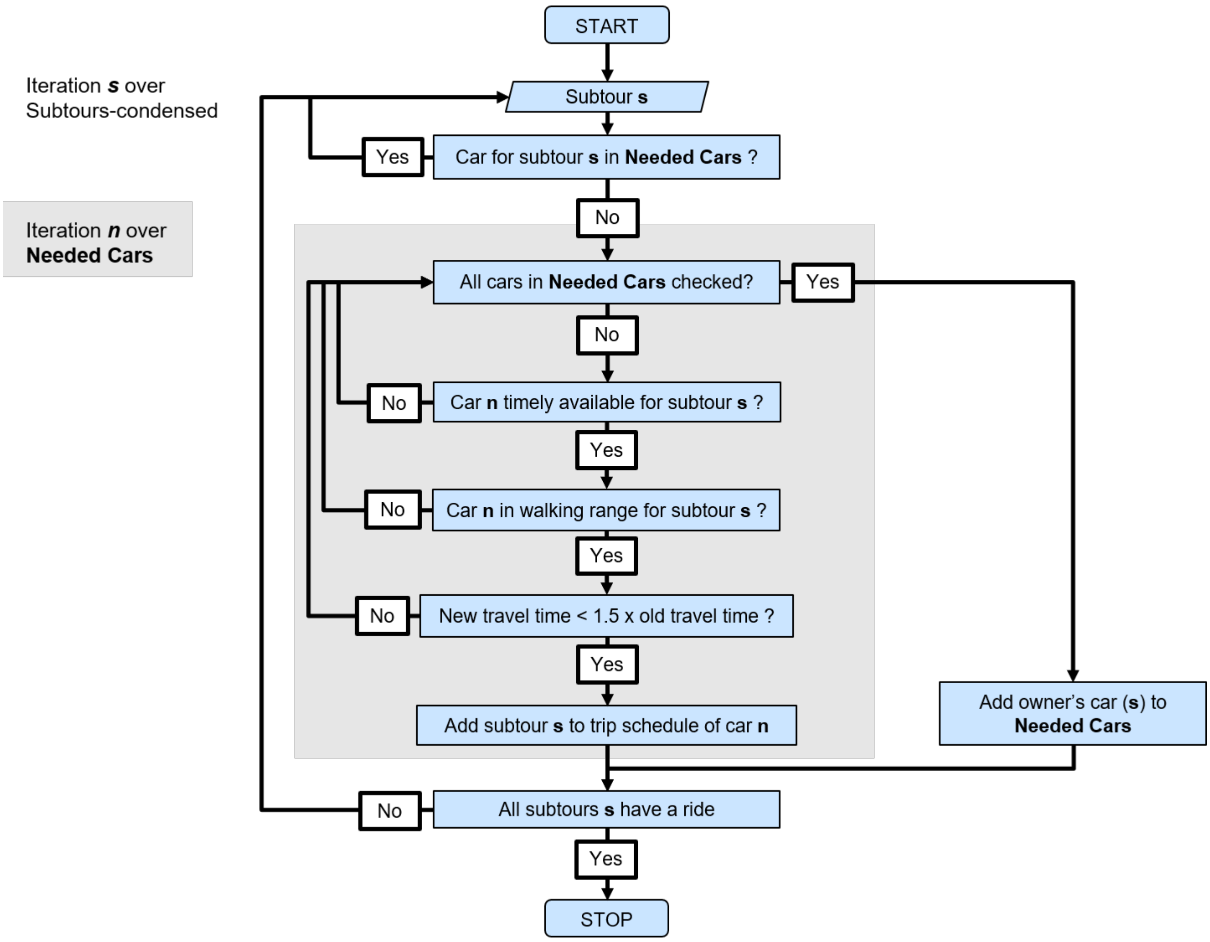
3.7.2 Implementation of Approach 2

The start for approach 2 is equal to Approaches 1.1 and 1.2. An outer loop introduces the subtour, for which we want to find a rental car. Different to the previous two approaches is the next step to check whether this subtour's agent has already used his car. In this case, of course, the representative takes his own car for the following subtours. Has the owner not used his car yet, there is still the chance of a possible rental and we go into an inner loop to check the 'needed-cars' and their compatibility with the subtour for a rental. After calculating the walking distance and checking time compatibility and walking range (500 m, 1,000 m, 1,500 m, 2,000 m), the rental car's schedule can either be updated because the rental takes place, or the inner loop finishes without a rental and the owner takes their own car for this and all following subtours. The 'needed-cars' contains now all the rental cars that are needed to fulfill all the subtours of the agents, where renting can only happen at the agent's home.

3.8 Approach 3

Approach 3 is the approach out of the three that comes closest to optimality. Additionally to approach 2, this approach also allows rentals to take part inbetween the owner's subtours. The greatest potential lies with working people who are at work from early morning until late at night and do not use their car inbetween. This approach tries to estimate this potential by implementing, that rentals can happen at other places than the initial location / the owner's home. For this, a similar data set to 'All-Subtours-condensed' Section 3.7 is defined - 'All-Trips-Schedule'. Since the rental is not restricted to one location, we cannot condense the trips of the agents to subtours, but have to implement all the trips separately in 'All-Trips-Schedule'. However, the idea is not to iterate over all those single trips, but to try and fit the subtours of 'Subtours-condensed' into the car trip schedules of the needed cars. As an illustration, the idea of Approach 3 is shown in Fig. 4.

Figure 4: Overview of Approach 3



3.8.1 All-Trips-Schedule (dictionary-type)

Before implementing Approach 3, 'All-Trips-Schedule' is filled with all the separate trip information of every agent. The goal is to establish a database to easily access a car's (or a car owner's) schedule. Moreover, the data set is based on the 'All-Trips' data set containing all the cleaned trips of every agent. By filtering 'All-Trips' for every agent, the whole schedule of the respective agent can be put into 'All-Trips-Schedule'. The included data is: starting time, ending time, initial vehicle location, final vehicle location, and the agent making the trip (the car owner).

In the end, 'All-Trips-Schedule' contains all agent's cars as entries with their whole schedule over the day. The schedule hereby does not consist of the roundtrips or subtours, but the individual trips from point A to point B.

3.8.2 Implementation of Approach 3

Approach 3 begins with the iteration over 'Subtours-condensed'. The aim is to search for rental cars (already used cars in 'needed cars', whose schedule allows for the subtour to take place inbetween two trips of this rental car. For this, a second loop is implemented inside the subtours-loop to look for a suitable car. The first step there is to check whether the subtour agent already used his car that day. While the first part is similar to the previous approaches, the second part is more complex and divided into three cases: (1) The subtour takes place before the rental car is used that day, (2) the subtour takes place after the rental car has made all its trips that day and (3) the subtour happen somewhere between the first and last trip of the car's schedule.

Case one checks, if the agent travels before the rental car is used. This is done by identifying the earliest trip of the rental car and verifying the condition that the analysed subtour's ending time plus the maximum allowed walking time to and from the rental location is smaller than the departure time of the earliest trip of the rental car. If this is the case, the actual additional travel time and walking distance are calculated. Given that the condition of the new travel time being less than 1.5 times the previous travel time, and the condition of the walking range are fulfilled, the rental takes place and the car schedule is updated to contain the information on the subtour.

Case two is very similar to the first case. While the subtour happens before the earliest trip in case one, case two considers the event that the subtour takes place only after the rental car has made its last trip of the day. Therefore, after locating the last trip of the rental car, the time compatibility is checked. When the condition is fulfilled, the walking distance and the new travel time are calculated to see if they fit the requirements. Their fulfillment leads to the rental and the rental car's schedule is updated with the subtour.

The **third case** is the most complex one. Whereas we only have to find the earliest or latest trip in the rental car's schedule, case three requires us to find the two trips between which the subtour takes place. There are two conditions that have to be met for the subtour to be timely compatible with the rental car: the subtour has to start only after the i-th trip ends, and the subtour has to end before trip i+1 starts. For the second condition, the ending time is adjusted by the maximum walking time to ensure the subtour can take place during the rental car's schedule gap. If the conditions are fulfilled, the actual additional travel time and walking distance is calculated. For the rental to be successful, the walking distance has to be within the predefined range (500 m, 1,000 m, 1,500 m, 2,000 m) and the new travel time cannot exceed 1.5 times the original travel time. After a successful rental, the rental car's schedule is updated.

With case three, the iteration over the rental car is complete. If there has not been a successful rental, the other rental cars are checked in the iteration. If no suitable rental car has been found, the agent has to use their own vehicle to make their subtour. This is implemented by adding the owner's car to the rental cars ('needed cars'). Now, all the following trips this agent makes are done by his own vehicle and his car is available for P2P users to rent.

Eventually, the 'needed cars' contains all the cars that are needed to fulfill the car trip demand, whereas the car owners can rent out their car not only at home, but also at their workplace.

4 Results

In this section, the results of Approach 3 are displayed. As Approaches 1.1, 1.2 and 2 do only show the process to building up to Approach 3, the results for these approaches can be found in Appendix A.1 and Appendix A.2.

Table 3: Number of trips in the utilized data sets

Data set	**Size of the data sets [number of trips]**					
Sample size	**1,000**	**10,000**	**50,000**	**100,000**	**200,000**	**500,000**
Share [%]	0.02	0.22	1.08	2.16	4.32	10.80
Filtered trips	363	3,350	16,700	33,200	66,300	166,000
All-Trips	137	1,460	7,590	15,200	30,600	76,000
Agents in All-Trips	48	487	2,520	5,000	10,100	25,200
Subtours	112	1,170	6,070	12,200	24,600	61,100
Subtours-condensed	56	585	3,040	6,100	12,300	30,700
Single-Subtours	96	974	5,030	9,990	20,300	50,400
Single-Subtours-condensed	48	487	2,520	5,000	10,100	25,200
All-Subtours-condensed	48	487	2,520	5,000	10,100	25,200
All-Trips-Schedule	48	487	2,520	5,000	10,100	25,200

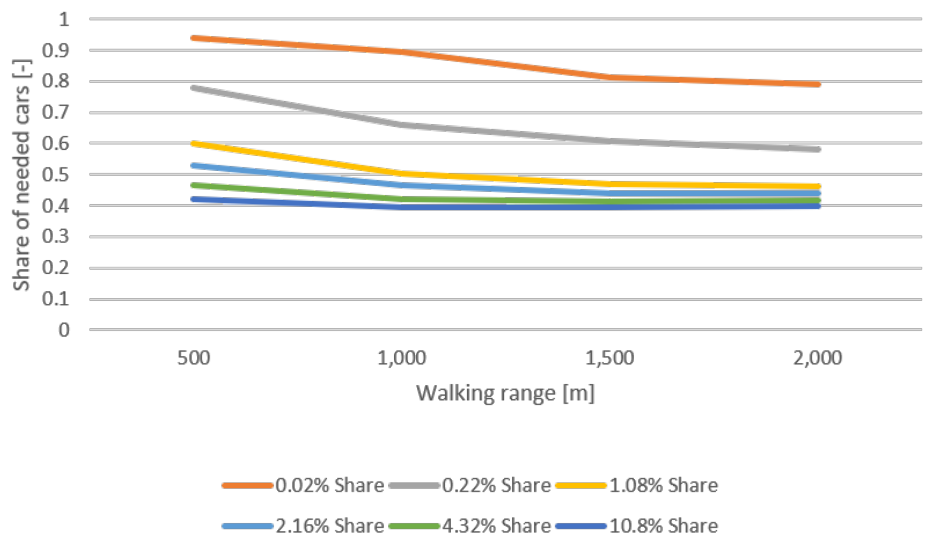
Table 3 shows the implemented shares of the initial data set (4.6M trips) and how many trips were left after the cleaning step and dropping the agents whose trips did not fulfill the boundary conditions. It is visible that almost all data sets were reduced by approximately 85% between the original sample size and the 'All-Trips' data set. As the number of agents is the same for Single-Subtours-condensed, All-Subtours-condensed and All-Trips-Schedule, it is reasonable that the sizes of the data sets are equal. The Subtours-condensed data frame has half as many trips as the Subtours data frame. This is explained by the fact, that two 'Subtours' trips are combined into one trip in the Subtours-condensed data frame.

Table 4: Number of needed cars to fulfill the car trip demand of Approach 3

Data set	**Size of the data sets [number of trips]**					
Sample size	**1,000**	**10,000**	**50,000**	**100,000**	**200,000**	**500,000**
Total cars	48	487	2,520	5,000	10,100	25,200
Walking range	**Number of needed cars**					
500 m	45	379	1,510	2,640	4,730	10,600
1,000 m	43	322	1,260	2,320	4,260	9,940
1,500 m	39	296	1,180	2,200	4,200	9,930
2,000 m	38	283	1,160	2,200	4,220	10,100

The quantitative results of Approach 3 with the different walking ranges and sample sizes are displayed in Table 4. With an increasing sample share, the number of total cars as well as the number of needed cars for the selected walking ranges increases. As expected, the factor between two sample shares is approximately the factor with which the number of total cars increases. The same applies for the number of needed cars with the different walking ranges. Furthermore, the number of needed cars decreases with increasing walking range, however, the rate of change does not appear to be linear, which indicates an inversely proportional relation. Interestingly, an anomaly can be detected for the 4.32% and the 10.8% sample share with a walking distance of 2,000 m. This could either be an error, or a hint, that the cars were shared differently, leading to more cars being needed overall. To illustrate the change of the share of needed cars in relation to the different sample shares and walking ranges, the results are displayed in Fig. 5.

Figure 5: Approach 3: Effect of different sample shares and walking ranges on the share of cars needed to satisfy the car trip demand

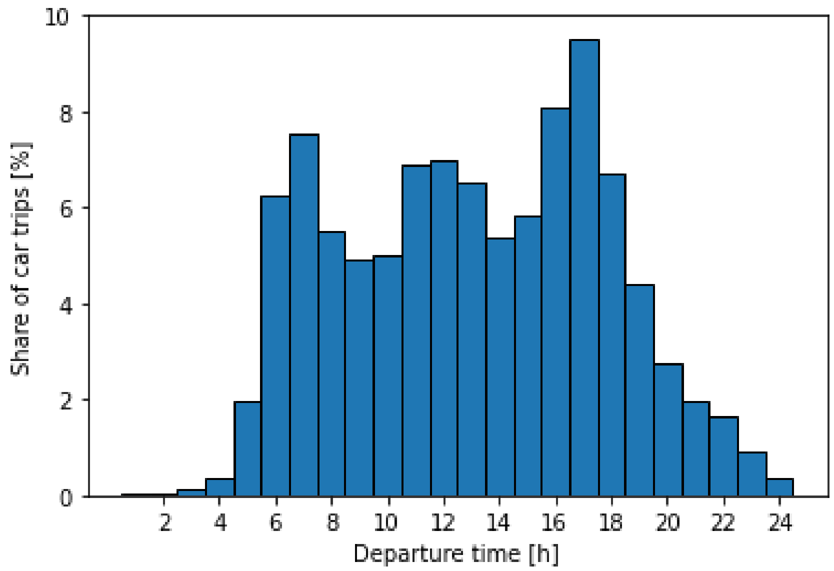


As can be seen in Fig. 5, the higher the share of the sample is, the smaller is the percentage of the total cars that is needed to satisfy the current car trip demand. With a share of 0.02%, almost all available cars are needed to satisfy the demand with a walking range of 500 m. When a walking distance of 2,000 m is allowed, however, the percentage of needed cars is reduced to appromately 85%. The greatest sample share shows a similar picture. A maximum walking distance of 500 m results in a 45% share of needed cars, whereas the extension of the walking range to 2,000 m decreases the share of needed cars to 40%. Therefore, the larger the share of the data set - the less impact the maximum walking distance has. However, it is important to note here that the influence of the walking range decreases with an increase in the sample share. This can be explained by the number of available rental locations which increases with the sample share. A greater sample share means more distributed and a denser net of vehicle locations, which results in more available vehicles in the agents' nearby area.

In the next subsections, the characteristics of the needed cars and their users will be analyzed. The characteristics analyzed include the departure times of the car trips, the trips per user, the number of users per needed car, the car trips per needed car, the rental locations and the walking distances to those locations, as well as the duration of the car trips. In order to keep the results concise and prevent repetitions, only the results with a 2,000 m walking range and the largest sample share will be displayed in these subsections. For completeness, the results with the largest sample share and the walking ranges of 500 m, 1,000 m and 1,500 m can be found in Appendix A.3.

4.1 Departure time of the car trips

Figure 6: Histogram showing the distribution of the departure times of the car trips

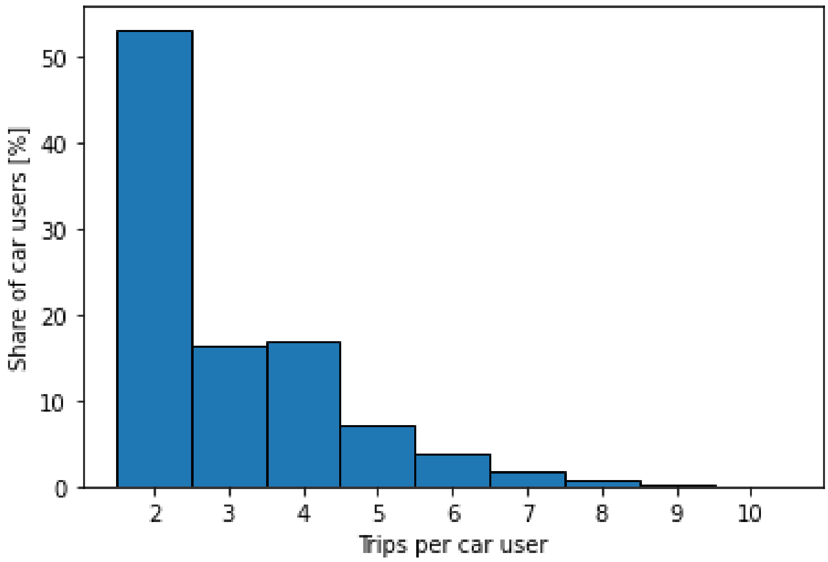


The histogram with the distribution of the departure times of the car trips is displayed in Fig. 6. Two peaks can be detected at 7 AM and at 5 PM. These are the rush hours, where most people leave for work or return back home. Another elevation can be discovered around lunchtime (approximately 12 PM). The broad distribution around this time frame could indicate that about 10% use their car to get lunch. What we can learn from this histogram is, that the greatest potential for P2P carsharing to happen is during the working hours, meaning from around 8 AM to 4 PM.

4.2 Trips per user

Looking at the number of trips per user (see Fig. 7), an evident high point can be observed at two trips per user. Assuming an average travel time of 30 minutes and that most people drive alone to work, this underlines the findings of Hampshire and Gaites (2011), stating that the cars sit idle for more than 90% of the day. In other words, it shows how underutilized those cars are. These results fall in line with the result, that only 40% of the cars are needed, because the cars are used very inefficiently and only for a short time every day, which allows a large rental time frame.

Figure 7: Histogram showing the distribution of the number of trips per user



4.3 Number of users per needed car

Figure 8: Histogram showing the distribution of the number of users per car with a maximum walking range of 2,000 m and a sample share of 10.8%

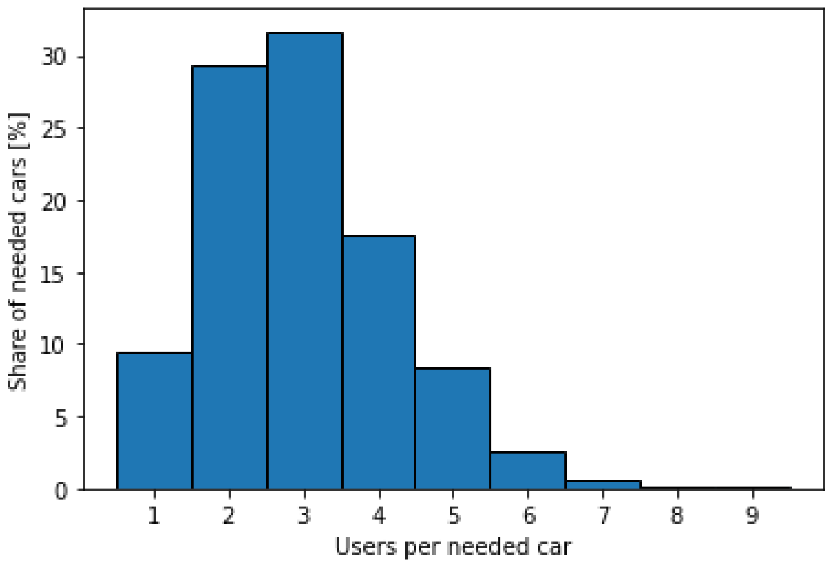
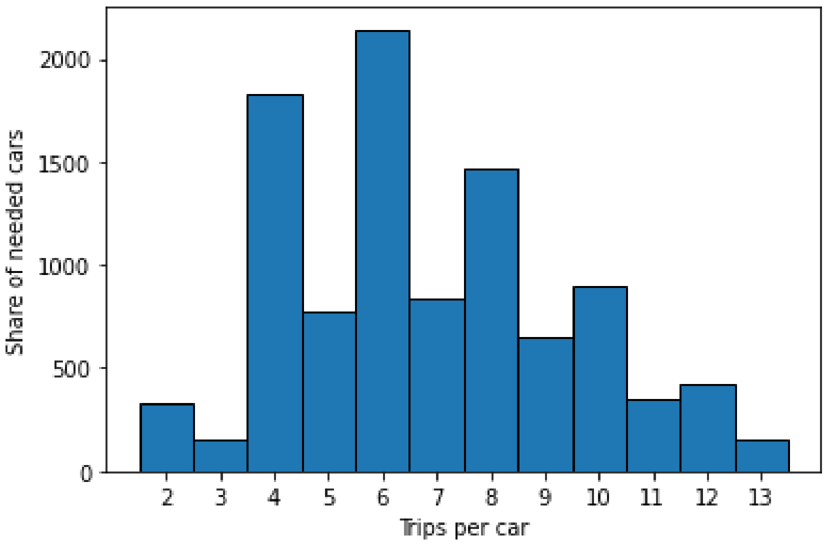


Fig. 8 displays the distribution of how many users drive around in the same needed car. Observing the histogram, we can see that the peak lies at two to three users per car and the mean is around three users. The fact that an average number of three users per car does not add up with 40% of the cars needed can be accounted for by users that use more than one car. This can be due to not finding a rental car for the second subtour, or just finding two different cars for two separate subtours. This leads to more needed cars than we expect with a mean of three users per car. In addition, looking at the proportion of 'one user per needed car', we can observe that the implementation of the P2P carsharing model has had a positive impact with almost tripling the trips the cars are needed for and, therefore, using them more efficiently.

4.4 Trips per needed car

Figure 9: Histogram showing the distribution of the number of trips per car with a maximum walking range of 2,000 m and a sample share of 10.8%

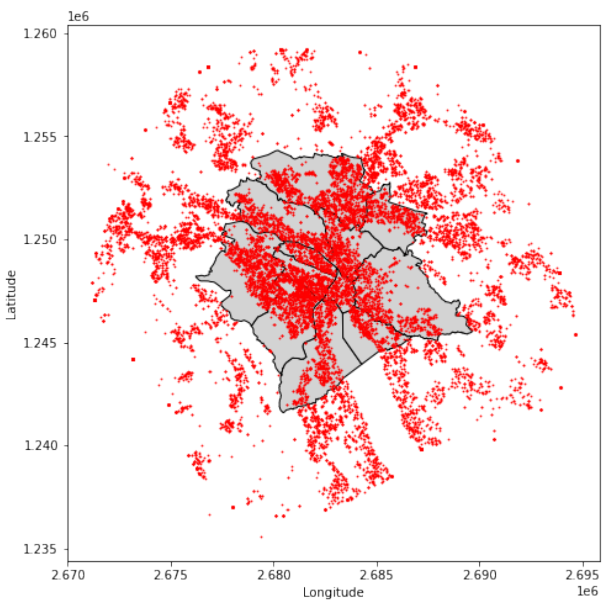


The distribution of the number of trips per car (see Fig. 9) shows a similar picture as the number of users per needed car. With a mean of 7 trips per car, around 3.5 users per needed car are expected, which is in conformity with the number of users per needed car (Fig. 8). In addition, one can see that the bars for 2-3 trips, meaning only one subtour per car, is very small. This indicates that only few single-subtour agents do not share their car. On the other hand, we cannot evaluate if agents using their car more often than for one subtour ($>$ 3 trips), do actually share their car. However, with the 1,000 users not having to share their car (Fig. 8) and only around 500 cars making only 2 or 3 trips (Fig. 9), it can be said that more than half of the users not sharing their car do travel for more than one subtour. Their schedule is, therefore, less likely to match with the schedule of other car users, because their car is already used multiple times. This means, their car is most likely used more efficiently than most other cars.

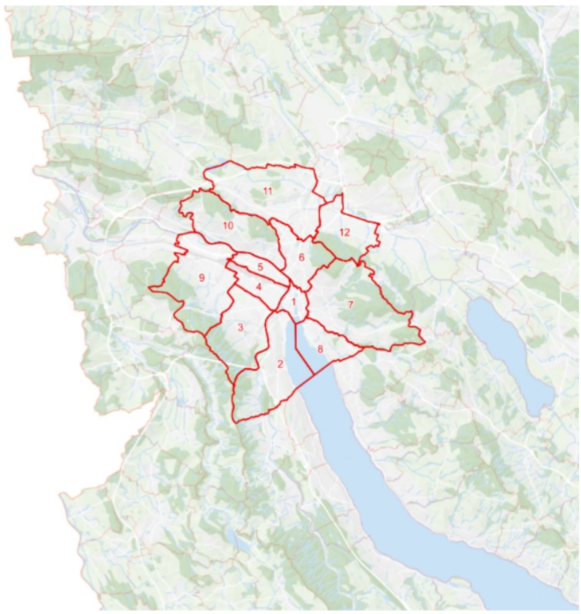
Furthermore, we detect lowpoints at an uneven number of trips per car. This can be explained with the average number of trips per user being two trips, and the fact that the rental subtours are all counted as two trips. This makes an odd trip number per car rather unlikely.

4.5 Rental locations

Figure 10: Scatter plot showing the rental locations with a maximum walking range of 2,000 m and a sample share of 10.8%



(a) Rental locations



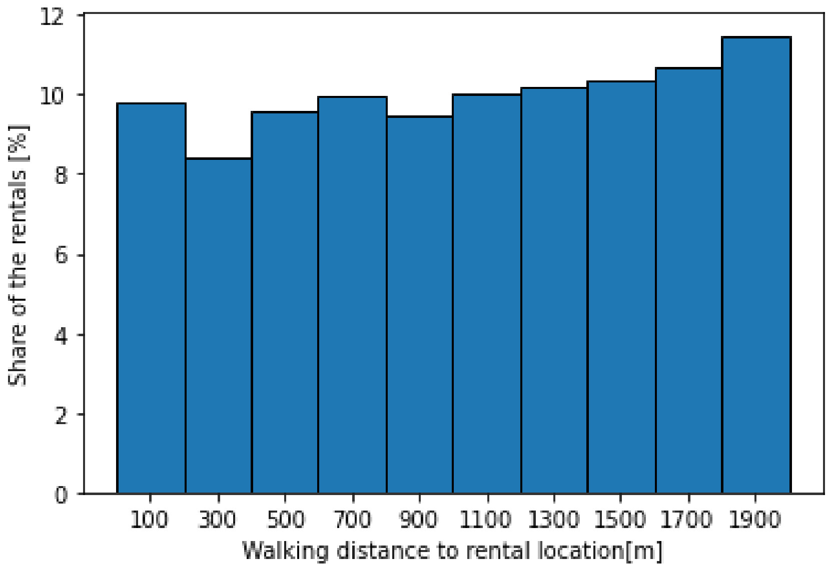
(b) Boundaries and city rings of the city of Zurich

Source: adopted from Stadt Zürich (2022)

When looking at the rental locations, the location of the city center becomes obvious (see Fig. 10). As we assumed that most rentals would take place at work, it makes sense for the rental locations to be concentrated in more dense areas - like the city center. The concentration of the rentals in the city center shows why traditional carsharing services focus on them (Hampshire and Gaites, 2011). Nonetheless, rentals happening outside the city center are still numerous and cannot be neglected, thus hinting at the profitability of P2P carsharing in more rural areas (Münzel *et al.*, 2018). This could be explained by people working outside the city center, or that the rentals took place at the owner's home. In woodlands and in rivers or lakes, there cannot be any rental locations, which is conformed with the scatter plot (Fig. 10(a)).

4.6 Walking distance to the rental location

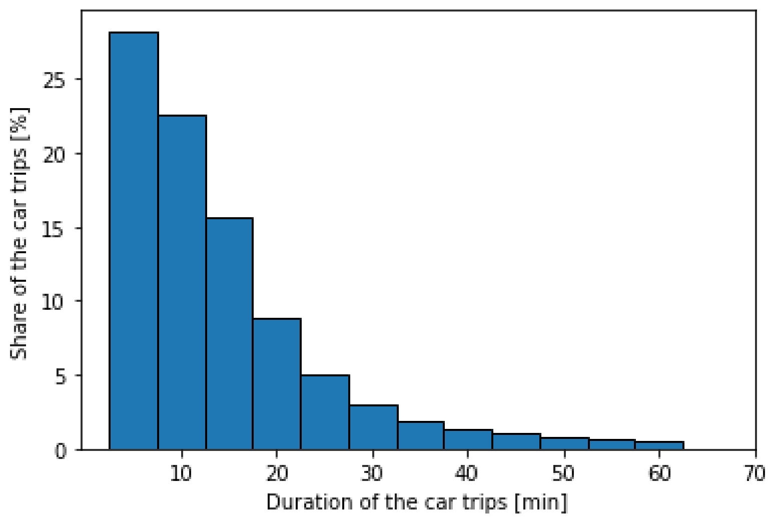
Figure 11: Histogram showing the walking distances to the rental locations with a maximum walking range of 2,000 m and a sample share of 10.8%



Considering the distribution of the walking distance to the rentals (Fig. 11), it can be discovered that the number of rentals is in positive relation to the walking distance. The most rentals happen at 2,000 m walking range, while lower walking ranges have fewer rentals. The number of rentals for the walking distance gives an explanation on why the share of needed cars gets smaller with a higher tolerance of the walking distance. However, it also underlines the findings of Fig. 5 showing the effect of the different walking shares and walking ranges on the share of needed cars, where an increased sample share was discovered to reduce the influence of the walking range on the share of needed cars. This is because the rate of increase of the rentals becomes less influential, the greater the number of rentals is, leading to little dependence of the share of needed cars on the walking range with increasing sample share.

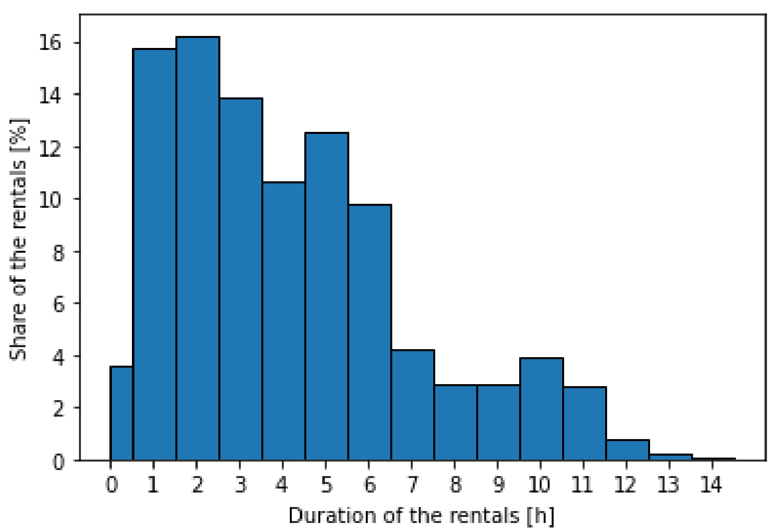
4.7 Duration of the car trips

Figure 13: Car trip duration with a sample share of 10.8%



This histogram (see Fig. 12) shows how long the car trips last. The histogram peaks at approximately five minutes and decreases rapidly with increasing trip duration. This further supports the findings of the cars sitting idle for 21 out of 24h by Hampshire and Gaites (2011) and shows how underutilized the cars are.

Figure 14: Histogram showing the duration of the rental subtours with a maximum walking range of 2,000 m and a sample share of 10.8%



The histogram depicting the rental durations Fig. 14 shows how large the schedule gap of some car owners is. With rental durations of up to over ten hours and a mean of 3.5 hours one can say that even for workers using their car to get lunch, P2P car sharing can be a viable option. Furthermore, in comparison with the individual trip durations Fig. 14 the longer subtour time can hint at the car being used for a longer period of time with rentals. However, this observation has to be interpreted with caution, since a rented car cannot be 'subrented' from a renter to another renter. In other words, the long rental hours might be due to the renter going to work later than the owner and the car being parked there instead of at the owner's workplace during the day.

5 Discussion and outlook

In this section, the implications and interpretations of the results will be discussed. Furthermore, the limitations of this research are addressed and recommendations for future research are presented.

5.1 Discussion of the results

The results showed that 40% of all available cars are enough to fulfill the current car trip demand in the greater area of Zurich. With an increasing sample share, it was observed that the share of needed cars decreased. This is in line with expectations, as a greater sample share equals more rental opportunities and more cars that are near the P2P member's location. It was also found, that a larger sample share meant less influence of the walking range. This can be explained by the distribution and density of the car's positions. If the sample share increases, there are more cars which will be distributed to the area. If it is now taken into account, that most people would rent out their car during work, it makes sense that there is a dense net of possible car rental locations in the city center, where people's workplace lies. Furthermore, if the density of the rental locations increases, this means that an overlapping of the cars' ranges becomes more likely and the probability rises that a possible rental car is near the P2P user's location. When we think about the convergence at 40%, this would mean, that more than half of all the cars are not needed. When looking at the departure times, one can see that not even half of the cars are used during peak hours of the departure times. This means that the result of 40% is plausible. Actually, it is rather surprising that the share of needed cars is not lower, when considering that the majority of the cars are only used for two trips a day. However, this might be explained by the fact, that people would have to walk to the car's location, which has to be in a 500 m, 1,000 m, 1,500 m or 2,000 m range. In addition, there are agents doing more than one subtour per day, which can lead to them taking a rental car for their first subtour, but having to use their own car for the second subtour. As could be seen in Fig. 11, the number of rentals increases linearly with the extension of the walking range. This can be explained by the fact, that no optimisation was applied to the problem. In an optimal approach, much less people would be willing to walk 2,000 m to the renting location of the car, resulting in an approximately inversely proportional shape of the distribution. Nonetheless, the implementation of the walking range condition makes the result more reasonable from a practical point of view, as it does account at least a little bit for the fact, that they only chose to rent when they feel the rental car is near enough. A second factor limiting the reduction of the needed cars is the condition, that the new travel time cannot exceed the old travel time by more than the factor of 1.5. Taking into account that a clear peak of the travel times is located at 10 minutes, it is also plausible that the share of needed cars is rather high at 40%, because an original travel time of 10 minutes would only allow for a walking distance of 160 m to the car, which limits the rental possibilities to a great extent.

When comparing the trips per user (or per car) before the P2P carsharing model was introduced into the equation and the trips per car after its introduction, we can clearly see the peaks shift from two trips per user and car to 4, 6 and 8 trips per car. Without the P2P model, the average number of trips per car lies around 2.2 trip, with the P2P model the average number of trips per car lies around 7.1. Although this does not take the travel time into account, it can be clearly said that the cars that are used more efficiently by a factor of more than 3. This would suggest, that the subtours per car increased from 1 to 3. This would then mean that we have an average of around 2-3 users per car. Fig. 8 supports this statement with a mean of around three users per needed car. Here we can also see, that there will still be cars which are needed to fulfill only one owner's trips. This can either be explained by the fact that the person travels only during peak hours when there's no timely available vehicle near, or that the person comes from the city and goes outside the city to work, so that the car is too far away from members wanting to rent a car, and so the car is not rented. Coming back to the limit of the new travel time, it can also be that the owner only travels with his car for less than 10 minutes and the possible walking range is too narrow for a car to be only about 100 m walking distance. It would also be highly possible that those users that do not share their needed cars are the ones which make 5-9 trips per day. If the vehicle owner uses their car themselves so many times, it would be difficult to find a time slot where a rental can happen, especially if those trips are done during the peak hours. Furthermore, if an owner uses their car for so many trips, the car's efficiency is already greater than for a majority of other cars, which is a positive aspect.

Another important point to discuss here, is the fact that the histogram showing the duration of the trips draws a slightly misleading picture. While at first sight, it could be assumed the cars are used the whole time during the rental duration, this is not the case. It is much more probable, that the renter themselves uses another's car to get to work and leaves the car at some parking for a long time while he is working. While this implementation is realistic, it is not realistic that a P2P member is prepared to pay the time he leaves the rented car at a parking space. The fact that the car cannot be 'subrented' might hint at a weakness of the P2P carsharing model, as a rented car would be too expensive for a short commute.

5.1.1 Positive impacts of Peer-to-Peer carsharing

One of the foremost positive impacts of the implementation of the P2P carsharing model is the reduction of the needed cars. While other carsharing businesses would have to buy new cars to extend their service, the P2P carsharing model uses already existing, but underused resources, which makes it much more sustainable. Although this cannot happen overnight, the fact that an alternative to buying a new car and using it very inefficiently exists and is adopted by many people (2EM, 2022b), is a step in the right direction to reducing the total number of cars around, and therefore, reducing the production and transport emission that incur because of the purchase of a new car. For those people who decide against buying a new car and for using a P2P service (young adults, people whose car has reached the end of its lifetime), this can be an opportunity to keep their environmentally friendly travel behavior or change their mode to a more sustainable and environmentally friendly one, e.g. public transport, biking.

Furthermore, for people that do not have the means to buy a car right now, the P2P carsharing can be a cost-efficient way of having access to a car whenever those people are in need of a car (Münzel *et al.*, 2019), e.g. grocery shopping, transport of large or heavy things. As Wilhelms *et al.* (2017) and Ballús-Armet *et al.* (2014) point out, with a P2P carsharing service it is much more likely to find a car fitting specific needs like a children's seat. This also means, that for each trip, the best car type option can be chosen. If you have your own car, you cannot adjust it to fit the needs of every kind of trip.

For the vehicle owner, the additional income can be a good motivation of putting up their vehicles on the platform. The result that around 3-25% of the interviewees are alone motivated by this additional revenue to take up carsharing (Hampshire and Gaites, 2011) shows the potential and the positive personal effects of making an effort of registering your car on a P2P website. These costs can then be used to either cover fixed costs of the cars or for other expenses or leisure activities. For ambitious P2P owners, the income could even be a catalyst to change their travel behaviour even just a little bit, by not using the car anymore for getting lunch to rent out their vehicle for a full day and not only a few hours.

As the results of the rental locations (see Fig. 10) implied, P2P carsharing has the potential of not only being profitable in an urban environment, but also in more rural areas. Though more research would have to be done on the potential of P2P carsharing in more rural areas to compare the two cases.

5.1.2 Negative impacts of Peer-to-Peer carsharing

On the other side of the coin, the availability of a cost-efficient alternative to owning a car could lead to ‘induced car usage’. This means, that people that have been using other modes of transportation might be tempted to use the car more often. With this, several negative effects occur. For one, the traffic on the streets and the CO_2 emissions increase. This is counterproductive for the goal of reducing the overall CO_2 emissions. Second, if people change their mode from public transport to a private vehicle, the sustainability of trains and busses is decreased, as unexploited capacity reduces the efficiency of the system and leads to larger emissions per person - thus decreasing its environmental friendliness, as we have seen these last few years happening due to the pandemic (SRF, 2022).

Another part that could discourage people from adopting P2P carsharing is the additional effort needed on both sides (vehicle owner and renter). The vehicle owner has to check with their insurance company whether the car has full coverage insurance, and has the burden of registering the car on the platform of the P2P service, which can take some time to do (Bollinger, 2022). If they do not want to install a keyless box, they need to be present for the handover of the keys and give some time for every rental. If the rental happens at the workplace, the company would have to show understanding for the worker to leave their work to complete the rent. In addition, the owner is responsible to keep their car clean and they cannot keep much of their personal belongings in their car. Furthermore, the owner gives up their flexibility / mobility, which restricts them greatly, in case they need to get somewhere on short notice.

Not only the vehicle owners, but also the vehicle renters have their comfort and flexibility reduced by giving up car ownership. A spontaneous trip with their own car is not possible (anymore), because each car trip has to be planned a few days beforehand. Firstly, the renter has to look for a vehicle to rent in a, for him, accessible location that is timely available. Next, the owner has to be contacted and has to accept the rental. If the owner accepts, the car has to be picked up and returned at a specific time. The need to plan every car trip can lead to considerable restrictions in flexibility. Especially if the person lives in a more rural area, where public transportation connections are limited.

5.2 Limitations

The primary factors that limit the scope of this research are the heuristic nature of the approach, the completeness or quality of the data set and the limited computing power. The heuristic nature of the approach is most noticeable when looking at the fact, that the first available rental option is taken, and no further car rental options are considered. In reality, the P2P member will choose the most optimal car for himself based on his needs, the walking distance and more factors. In other words, the arbitrariness of the renters cannot be profoundly accounted for in a heuristic approach. In addition, it is also neglected that there is a 'best option' for the whole system. This means, we do not care that there is a different rental option B that, in the end, requires fewer cars for the system, than if we decide to take option A for this rental. Another important point to note here, is that the cars cannot be 'subrented'. This especially distorts the picture drawn by the duration of the travel times, where the rentals are considered as whole car trip durations.

An incompleteness or poor quality of the data set leads to a great reduction in the already rather small sample share. As seen in Table 3, the original sample size has been further reduced to only around 15%. The analysis of only a small proportion of the scenario leads to an extrapolated interpretation, which means we conclude implications from a small sample to a greater set of data. The problem is, that with this extrapolation comes great uncertainty, as the investigated sample is not necessarily (or cannot be) representative of the whole data.

An example for the problem with extrapolation is the anomaly in Table 3, where the number of needed cars was higher for the most extensive walking range of 2,000 m with the highest two sample shares. On one hand, there is a clear convergence around 40%, but on the other hand, the highest two sample shares show a higher number of cars, although we would expect there to be less than with the 1,500 m walking range. As the code remained almost the same, it can be argued that the different walking range distributed the rentals differently - thus leading to a 'less efficient' solution with more needed cars.

The uncertainty is further compounded by a limiting computer power. Although the data set had an original trip size of 4.6M, the greatest input amounted only to 500,000 trips, which is approximately 10% of the data set. In the end, this leads to the analysis of only a very small portion of the scenario. On the other hand, we saw that the doubling of the sample share from 4.32% to 10.8% does only result in an average change of 3% in the share of needed cars (see Fig. 5). This illustrates, that the convergence of the small sample share shows consistency and that the doubling of the sample share leads to only a small difference in the share of needed cars. It can, therefore, also be argued, that a higher computing power would not result in a much lower share of needed cars and would not give much new insight considering the other uncertainties in the calculation that resulted mainly because of the heuristic nature. However, there still remains the possibility that the sample is absolutely not representative of all the activities of the agents in the analysed MATSim scenario, which would reduce the significance of the result to a great extent.

5.3 Recommendations / Outlook

For further investigation, it would be interesting to look at the potential of P2P carsharing in a more rural area. For the future expansion of the P2P carsharing model it would be helpful to get an idea of the usefulness of this service in areas, where traditional carsharing cannot expand to. As Münzel *et al.* (2018), van der Linden (2016), Hampshire and Gaites (2011) and Shaheen *et al.* (2012) explained, the agnostic nature of the P2P model could increase the flexiblity and accessibility of the people not having sufficient public transportation connections and motivate them to share their car with their neighbours.

Another focus that would be of interest is the estimation of the attractiveness of the P2P model. Although this thesis has not integrated the attractiveness of P2P in a very pronounced way, it would be of great interest to capture the probabilities with which the P2P service would be chosen. With the implementation of a behavioural logit model, the decisive factors for the adoption of P2P membership could be explored and used to increase the competitiveness of this relatively new mode of transportation.

To extend this research in particular, the analysis of different penetration rates of car owners willing to rent out their car would give useful insights into the P2P model's market potential and could help to get a more accurate share of how many cars will actually be needed to satisfy the current car trip demand.

6 Conclusion

With the prevalent issue of climate change and the reduction of CO_2 emissions dominating many discussions, it is important to focus on how to make better use of our already existing resources. P2P starts exactly there and focuses on using our underused vehicles more efficiently. In an approach to gauge the potential of this service, this thesis aimed to estimate how many vehicles could satisfy the current car trip demand in the greater area of Zurich, Switzerland. For this, a heuristic approach was implemented to analyse the activities in a MATSim scenario containing boundaries of the city of Zurich and a 5 km buffer zone. The implementation followed the general idea of finding subtours which can fit into the schedule of a car that has already been used by its owner that day. For a rental, three conditions had to be fulfilled: First, the car's location had to be in a walking range to the initial position of the renter of either 500 m, 1,000 m, 1,500 m or 2,000 m. Second, the owner's travel has priority, meaning the renter can only rent the car while the owner does not need the car. Third, the new travel time with the rental cannot exceed 1.5 times the old travel with the user's own car.

The results yielded a share of needed cars of approximately 40%. Hereby, the greatest potential was found to be during the working hours from 8 AM to 4 PM. The calculated share of 40% has to be interpreted as a reference value, as several factors that limit the potential were neglected, such as the fact, that not all car owners will participate in a P2P carsharing service and that the additional effort will not be worth it for some car users. On the other hand, the additional income generated by the vehicle owners can not only cover some of their fixed costs, but can also motivate ambitious owners to use other modes of transportation in order to rent out their car for a full working day rather than only a few hours. In addition, a cost-efficient alternative to car ownership can extend the mobility of people with limited means and allow them the flexibility of having a car fitting their needs when they are in need of one. The negative side of this is, however, that other mode users (public transportation, bike, etc.) will be tempted to now use a car, resulting in induced car usage. This contradicts the goal of the P2P carsharing model and leads to more traffic, more unexploited capacity in public transportation and, therefore, an increase in emissions. Nevertheless, the results of this thesis showed that the number of users and trips per car can be (more than) tripled and that great potential for this service lies at city centers, although P2P also fulfills the requirements to be profitable even in more rural areas.

7 References

2EM (2022a) 2EM Car Sharing ist nun die einzige Plattform für Carsharing unter Privatpersonen in der Schweiz, March 2022, https://www.2em.ch/presse/articles/transition-sharoo-de.pdf.

2EM (2022b) Car rental among private individuals in Switzerland, webpage, March 2022, https://de.2em.ch/.

Ballús-Armet, I., S. A. Shaheen, K. Clonts and D. Weinzimmer (2014) Peer-to-Peer carsharing: Exploring public perception and market characteristics in the San Francisco Bay Area, California, *Transportation Research Record*, **2416**, 27–36.

Baloise (2022) GoMore - Burcu Biçer, Marktverantwortliche Schweiz, im Interview, April 2022, https://www.baloise.com/de/home/news-stories/news/blog/2022/gomore-burcu-bicer-marktverantwortliche-schweiz-im-interview.html.

Barbour, N., Y. Zhang and F. Mannering (2020) Individuals' willingness to rent their personal vehicle to others: An exploratory assessment of peer-to-peer carsharing, *Transportation Research Interdisciplinary Perspectives*, **5**, 100138.

Benjaafar, S., G. Kong, X. Li and C. Courcoubetis (2019) Peer-to-peer product sharing: Implications for ownership, usage, and social welfare in the sharing economy, *Management Science*, **65** (2) 477–493.

Bollinger, T. (2022) Technikblog - Privates Carsharing im Vergleich: Vermieten über 2em oder Gomore?, April 2022, https://technikblog.ch/2022/03/privates-carsharing-im-vergleich-vermieten-ueber-2em-oder-gomore/.

Bundesamt für Statistik (2022) Umweltauswirkungen, April 2022, https://www.bfs.admin.ch/bfs/de/home/statistiken/mobilitaet-verkehr/unfaelle-umweltauswirkungen/umweltauswirkungen.html.

CERTU - CETE de Lyon (2011) Note sur l'Autopartage de voisinage - Etat de l'art en Europe et Amérique du Nord, March 2022, http://www.adetec-deplacements.com/etat-de-l-art-AEP-Europe-Am%E9rique-Nord.pdf.

Chen, T. D. and K. M. Kockelman (2016) Carsharing's life-cycle impacts on energy use and greenhouse gas emissions, *Transportation Research Part D: Transport and Environment*, **47**, 276–284.

Dill, J., N. McNeil and S. Howland (2019) Effects of peer-to-peer carsharing on vehicle owners' travel behavior, *Transportation Research Part C: Emerging Technologies*, **101**, 70–78.

EPFL (2022) Noto - EPFL's JupyterLab, webpage, June 2022, `http://noto.epfl.ch`.

European Parliament (2022) CO_2 emissions from cars: facts and figures (infographics), April 2022, `https://www.europarl.europa.eu/news/en/headlines/society/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics`.

Flick, C. and C. Henseling (2019) Plattformen des Peer-to-Peer Sharing, *Digitale Kultur des Teilens: Mit Sharing nachhaltiger Wirtschaften*, 35–39, Springer Fachmedien Wiesbaden, Wiesbaden.

GoMore (2022) Private car rental and leasing, webpage, March 2022, `https://gomore.ch/`.

Hampshire, R. C. and C. Gaites (2011) Peer-to-Peer carsharing: Market analysis and potential growth, *Transportation Research Record*, **2217**, 119–126.

Harms, S. and B. Truffer (1998) The emergence of a nation-wide carsharing co-operative in Switzerland, *Technical Report*, EAWAG, University of Twente, AE Enschede, Netherlands.

Horni, A., K. Nagel and K. W. Axhausen (eds.) (2016) *The Multi-Agent Transport Simulation MATSim*, Ubiquity Press, London.

Hörl, S. (2020) Dynamic demand simulation for automated mobility on demand, Ph.D. thesis, ETH Zurich, Zurich.

Hörl, S., F. Becker, T. J. P. Dubernet and K. W. Axhausen (2018) Induzierter Verkehr durch autonome Fahrzeuge: Eine Abschätzung, *Final Report for SVI 2016/001*, Bundesamt für Strassen (ASTRA), Ittingen.

Loose, W. (2016) Mehr Platz zum Leben – wie CarSharing Städte entlastet, *Endbericht*

„CarSharing im innerstädtischen Raum – eine Wirkungsanalyse", Bundesverband Carsharing, Berlin.

Millard-Ball, A., G. Murray, J. T. Schure, C. Fix and J. Burkhardt (2005) Car-sharing: Where and how it succeeds, *Technical Report*, **108**, Washington D.C.

Mobiliar (2022) Das Auto des Nachbarn fahren, April 2022, `https://www.mobiliar.ch/die-mobiliar/medien/medienmitteilungen/das-auto-des-nachbarn-fahren`.

Mobility Carsharing (2022) Carsharing operator (B2C, station-based), webpage, April 2022, `https://www.mobility.ch/`.

Münzel, K., W. Boon, K. Frenken and T. Vaskelainen (2018) Carsharing business models in Germany: Characteristics, success and future prospects, *Information Systems and e-Business Management*, **16** (2) 271–291.

Münzel, K., L. Piscicelli, W. Boon and K. Frenken (2019) Different business models – different users? Uncovering the motives and characteristics of business-to-consumer and peer-to-peer carsharing adopters in The Netherlands, *Transportation Research Part D: Transport and Environment*, **73**, 276–306.

Pandas (2022) Open source data analysis and manipulation tool, built on top of the Python programming language, webpage, June 2022, `https://pandas.pydata.org/`.

Rideable (2022) Sharoo gibt auf - neue Mobility Konkurrenz in den Startlöchern, April 2022, `https://rideable.ch/sharoo-gibt-auf-neue-mobility-konkurrenz-in-den-startlochern/`.

Shaheen, S. and A. Cohen (2020) Innovative Mobility: Carsharing outlook; carsharing market overview, analysis, and trends, *Transportation Sustainability Research Center*, University of California, Berkeley, Berkeley (USA).

Shaheen, S., E. Martin and A. Bansal (2018) Peer-To-Peer (P2P) carsharing: Understanding early markets, social dynamics, and behavioral impacts, *Transportation Sustainability Research Center*, University of California, Berkeley, Berkeley (USA).

Shaheen, S. A., M. A. Mallery and K. J. Kingsley (2012) Personal vehicle sharing services in North America, *Research in Transportation Business & Management*, **3**, 71–81.

SRF (2022) ÖV leidet massiv unter den Folgen der Pandemie, June 2022, https://www.srf.ch/news/schweiz/neue-zahlen-zur-mobilitaet-oev-leidet-massiv-unter-den-folgen-der-pandemie.

Stadt Zürich (2022) Geodaten Stadtkreise, May 2022, https://www.stadt-zuerich.ch/geodaten/download/Stadtkreise?format=10008.

Tagesanzeiger (2022) Fremde in meinem Auto? Nein danke!, April 2022, https://www.tagesanzeiger.ch/wirtschaft/unternehmen-und-konjunktur/fremde-in-meinem-auto-nein-danke/story/28247458.

van der Linden, D. F. (2016) Explaining the differential growth of peer-to-peer car-sharing in European cities, Master thesis, Utrecht University, Utrecht, Netherlands.

Wilhelms, M.-P., S. Henkel and K. Merfeld (2017) You are what you share: Understanding participation motives in peer-to-peer carsharing, in: Meyer G., Shaheen S. (Eds), Disrupting Mobility, Springer, Cham, 105–119.

A Appendix

A.1 Trips in the created data sets

Table 5: Number of trip in the created data sets

Data set	**Size of the data sets [number of trips]**					
Sample size	**1,000**	**10,000**	**50,000**	**100,000**	**200,000**	**500,000**
Sample share [%]	0.02	0.22	1.08	2.16	4.32	10.80
Filtered trips	363	3,352	16,747	33,199	66,327	165,866
All-Trips	137	1,464	7,586	15,153	30,639	76,049
Agents in All-Trips	48	487	2,516	4,995	10,129	25,215
Subtours	112	1,170	6,072	12,192	24,576	61,098
Subtours-condensed	56	585	3,036	6,100	12,294	30,650
Single-Subtours	96	974	5,032	9,990	20,258	50,430
Single-Subtours-condensed	48	487	2,516	4,995	10,129	25,215
All-Subtours-condensed	48	487	2,516	4,995	10,129	25,215
All-Trips-Schedule	48	487	2,516	4,995	10,129	25,215
500 m walking range						
Needed-cars-1-1	47	398	1,676	3,030	5,556	10,992
Needed-cars-1-2	47	408	1,789	3,387	6,568	15,727
Needed-cars-2	47	429	1,970	3,761	7,350	17,774
Needed-cars-3	45	379	1,513	2,642	4,732	10,628
1,000 m walking range						
Needed-cars-1-1	46	347	1,431	2,549	4,461	7,589
Needed-cars-1-2	46	368	1,634	3,171	6,276	15,371
Needed-cars-2	47	394	1,817	3,566	7,091	17,503
Needed-cars-3	43	322	1,261	2,317	4,259	9,938
1,500 m walking range						
Needed-cars-1-1	43	323	1,292	2,232	3,744	5,922
Needed-cars-1-2	43	346	1,586	3,106	6,183	15,260
Needed-cars-2	45	374	1,767	3,513	7,037	17,407
Needed-cars-3	39	296	1,177	2,203	4,204	9,934
2,000 m walking range						
Needed-cars-1-1	41	305	1,162	1,983	3,230	4,792
Needed-cars-1-2	41	335	1,560	3,064	6,135	15,183
Needed-cars-2	43	363	1,761	3,487	6,998	17,327
Needed-cars-3	38	283	1,159	2,195	4,221	10,083

A.2 Results of Approaches 1.1, 1.2, and 2

Figure 15: Approach 1.1: Effect of different sample shares and walking ranges on the share of cars needed to satisfy the car trip demand

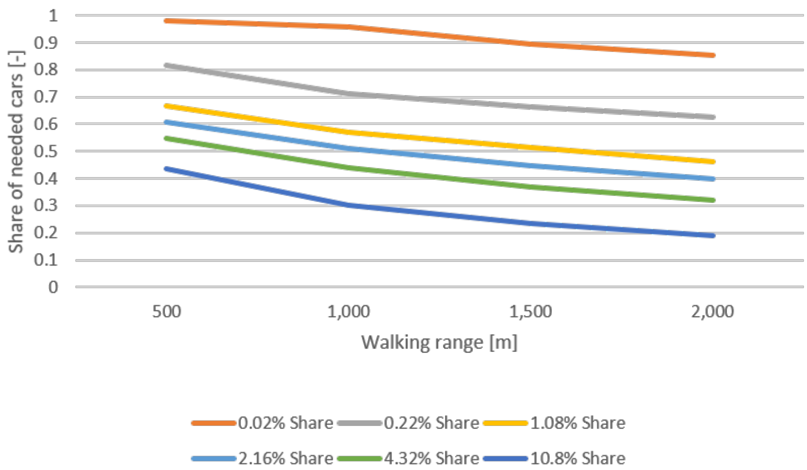


Figure 16: Approach 1.2: Effect of different sample shares and walking ranges on the share of cars needed to satisfy the car trip demand

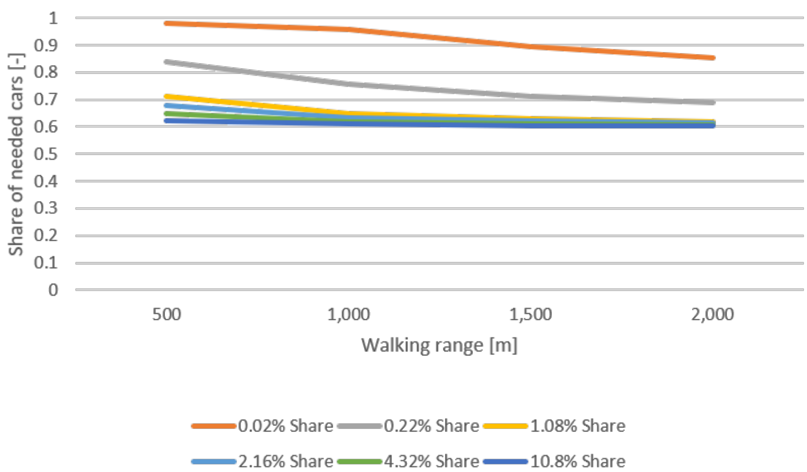
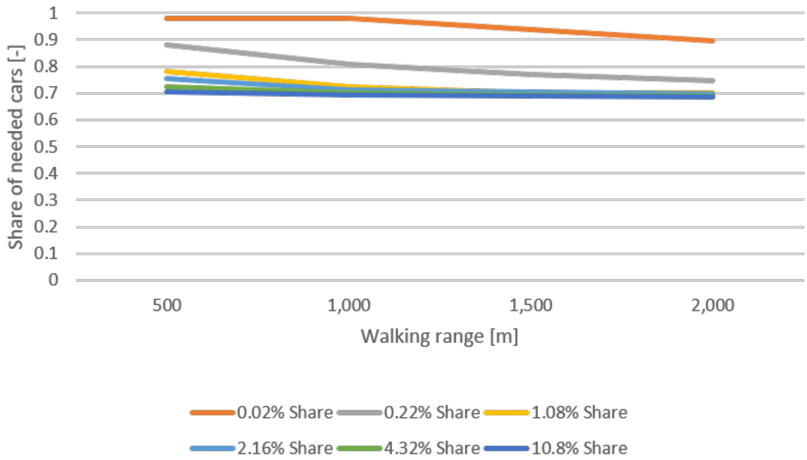


Figure 17: Approach 2: Effect of different sample shares and walking ranges on the share of cars needed to satisfy the car trip demand



A.3 Histograms of 500 m, 1,000 m, and 1,500 m walking range

A.3.1 Histograms showing the users per needed car

Figure 18: Histogram showing the distribution of the number of users per car with a maximum walking range of 500 m and a sample share of 10.8%

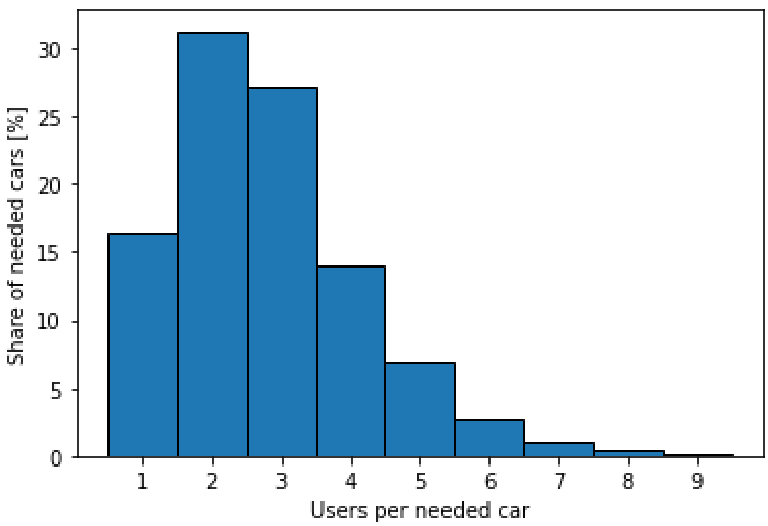


Figure 19: Histogram showing the distribution of the number of users per car with a maximum walking range of 1,000 m and a sample share of 10.8%

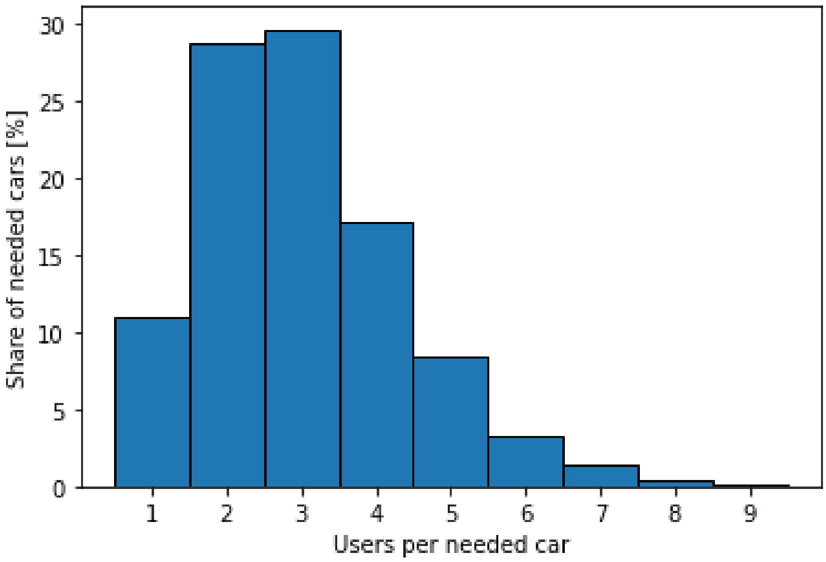
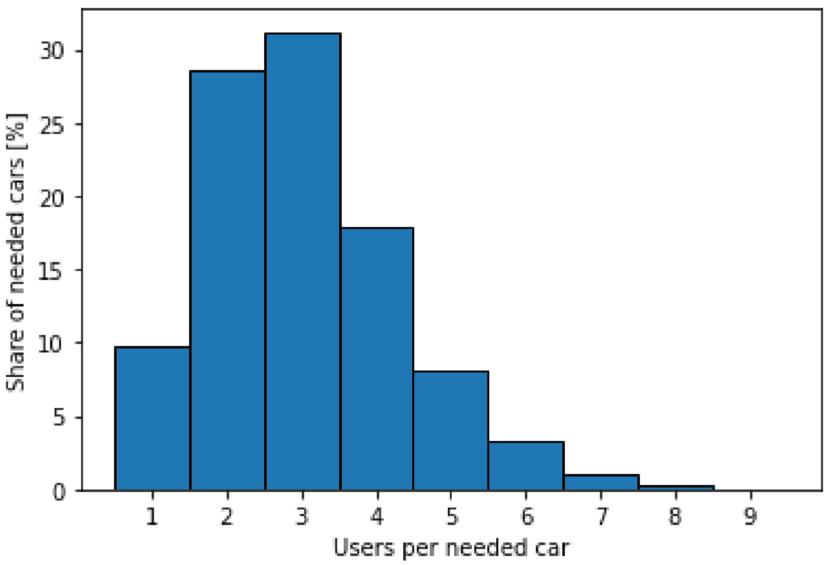


Figure 20: Histogram showing the distribution of the number of users per car with a maximum walking range of 1,500 m and a sample share of 10.8%



A.3.2 Histograms showing the trips per needed car

Figure 21: Histogram showing the distribution of the number of trips per car with a maximum walking range of 500 m and a sample share of 10.8%

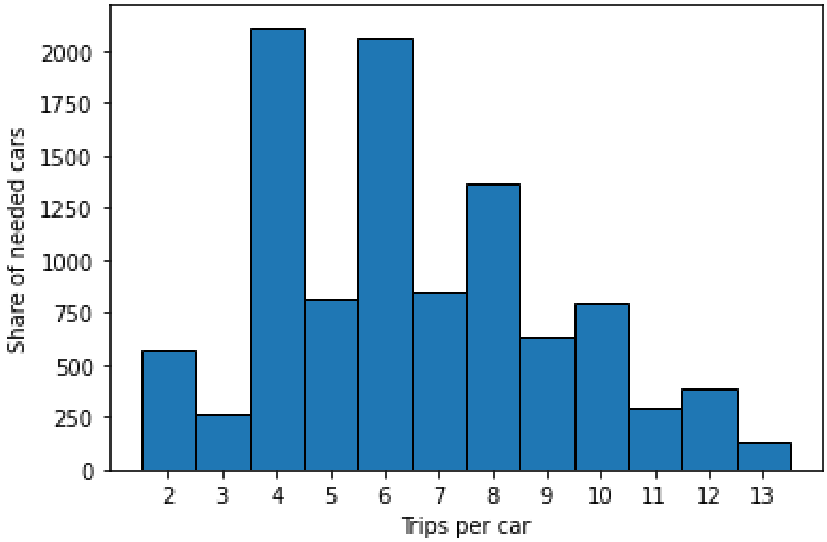


Figure 22: Histogram showing the distribution of the number of trips per car with a maximum walking range of 1,000 m and a sample share of 10.8%

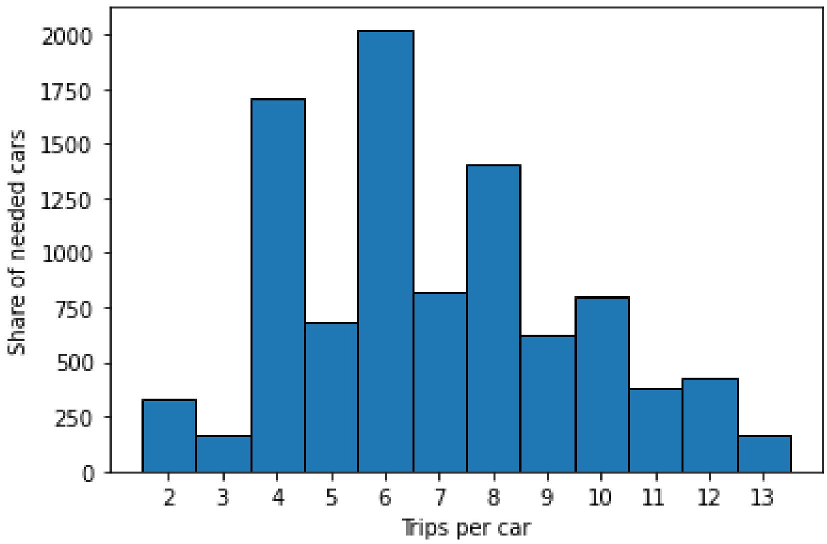
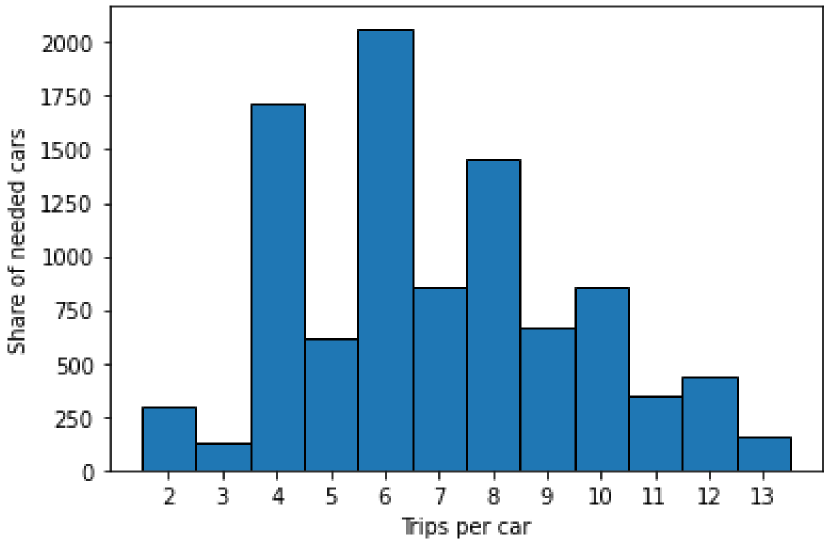
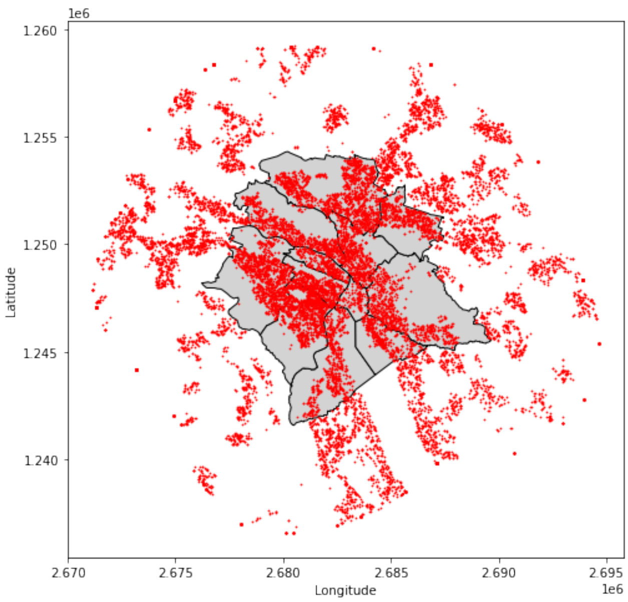


Figure 23: Histogram showing the distribution of the number of trips per car with a maximum walking range of 1,500 m and a sample share of 10.8%



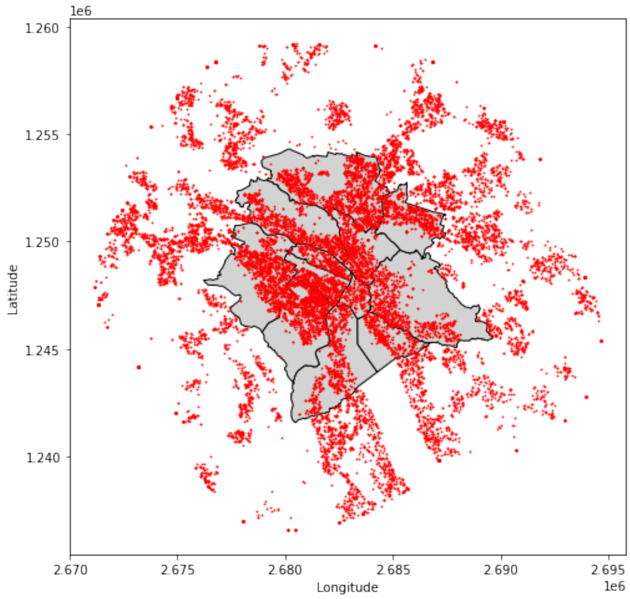
A.3.3 Scatter plot showing the rental locations

Figure 24: Scatter plot showing the rental locations with a maximum walking range of 500 m and a sample share of 10.8%



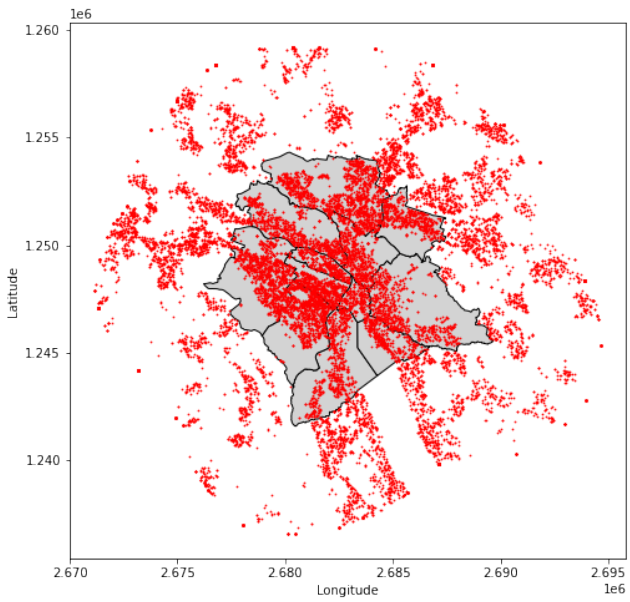
Source: adopted from Stadt Zürich (2022)

Figure 25: Scatter plot showing the rental locations with a maximum walking range of 1,000 m and a sample share of 10.8%



Source: adopted from Stadt Zürich (2022)

Figure 26: Scatter plot showing the rental locations with a maximum walking range of 1,500 m and a sample share of 10.8%



Source: adopted from Stadt Zürich (2022)

A.3.4 Histograms showing the walking distance to the rental location

Figure 27: Histogram showing the walking distances to the rental locations with a maximum walking range of 500 m and a sample share of 10.8%

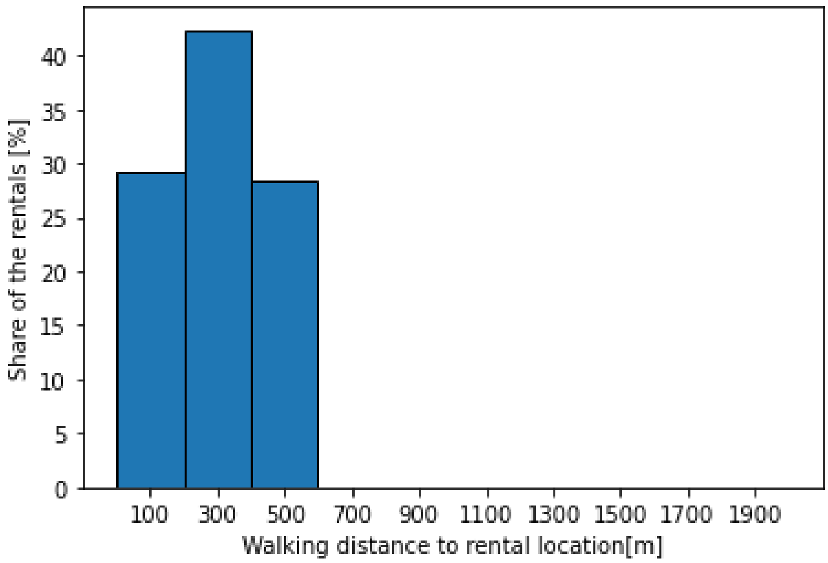


Figure 28: Histogram showing the walking distances to the rental locations with a maximum walking range of 1,000 m and a sample share of 10.8%

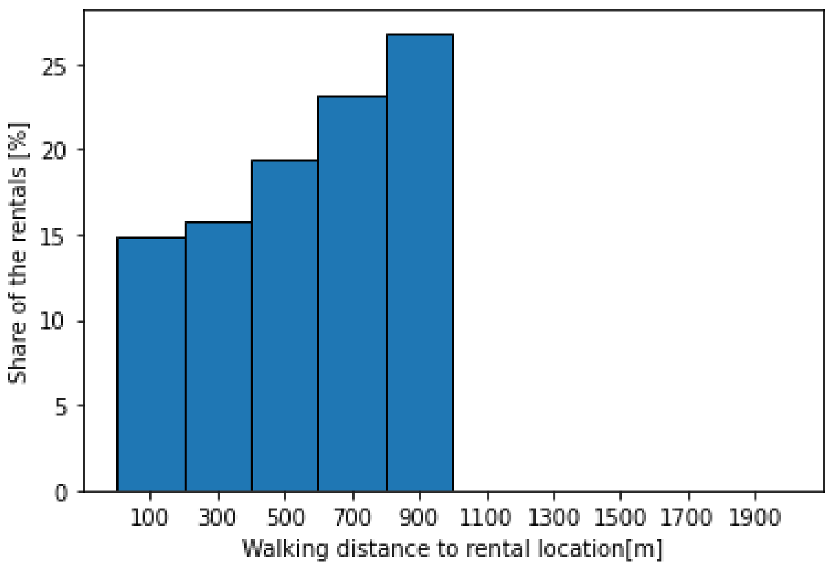
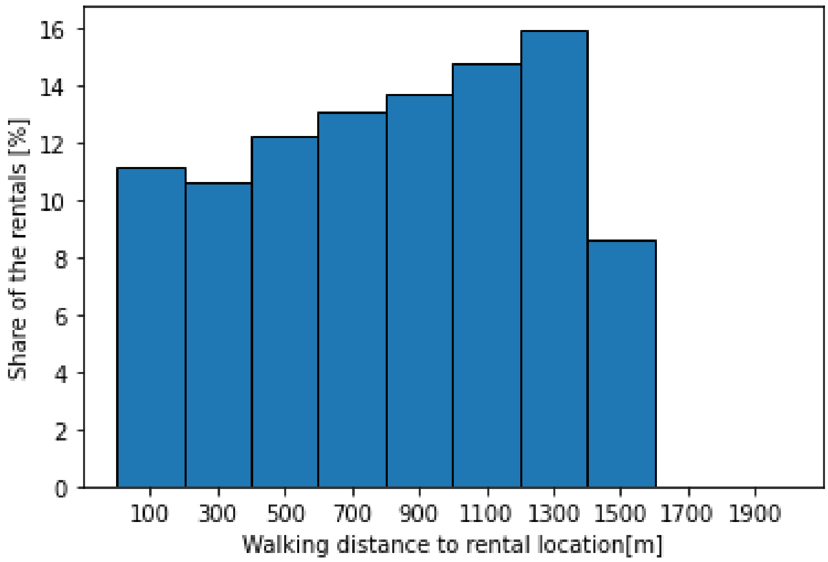


Figure 29: Histogram showing the walking distances to the rental locations with a maximum walking range of 1,500 m and a sample share of 10.8%



A.3.5 Histograms showing the trip duration of the rentals

Figure 31: Trips' duration of the needed cars with a maximum walking range of 500 m and a sample share of 10.8%

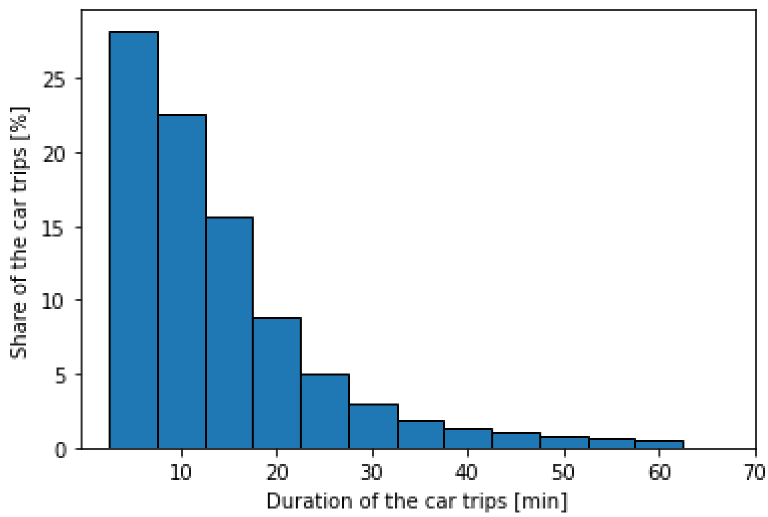


Figure 32: Histogram showing the duration of the rental subtours with a maximum walking range of 500 m and a sample share of 10.8%

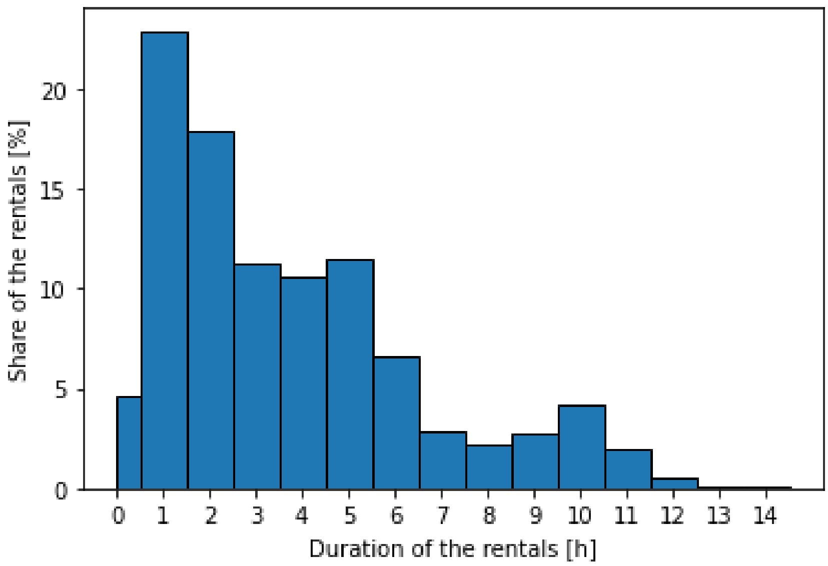


Figure 34: Trips' duration of the needed cars with a maximum walking range of 1,000 m and a sample share of 10.8%

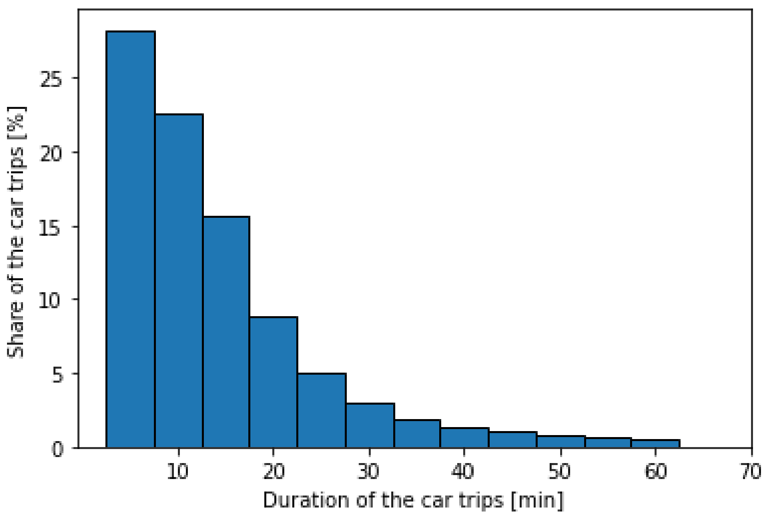


Figure 35: Histogram showing the duration of the rental subtours with a maximum walking range of 1,000 m and a sample share of 10.8%

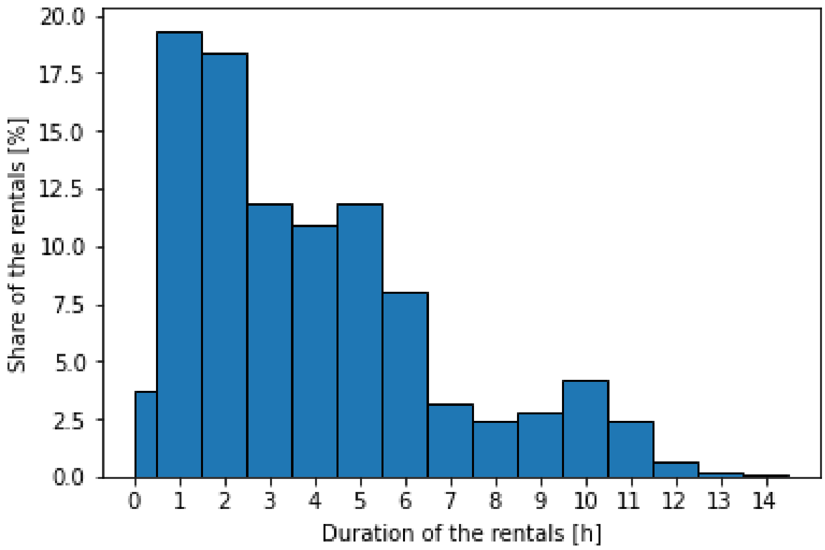


Figure 37: Trips' duration of the needed cars with a maximum walking range of 1,500 m and a sample share of 10.8%

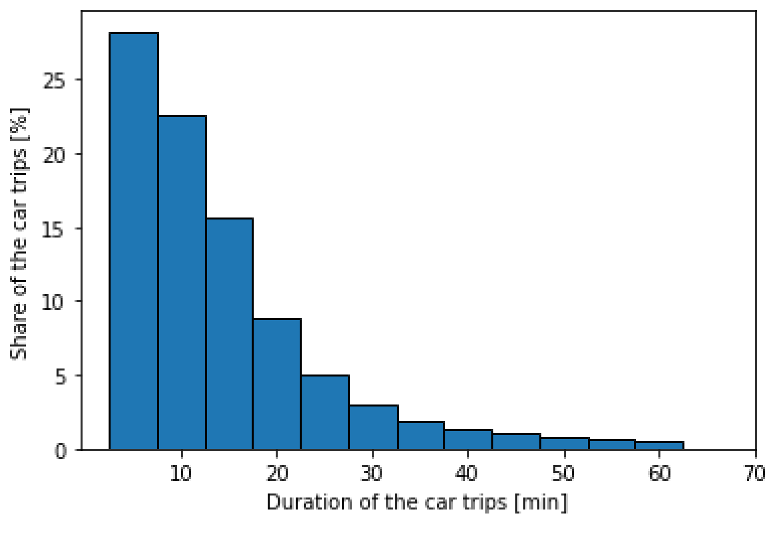
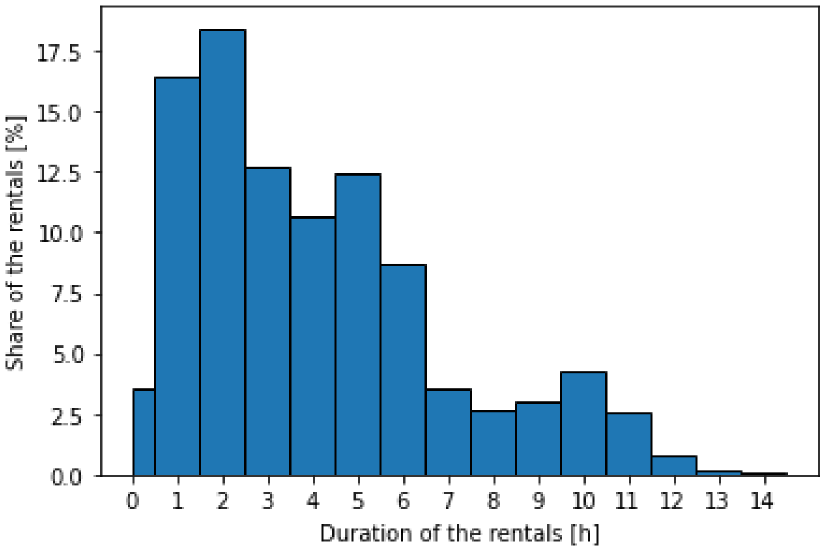


Figure 38: Histogram showing the duration of the rental subtours with a maximum walking range of 1,500 m and a sample share of 10.8%



A.4 Python Code for the implementation of the approaches

Figure 39: Python Code for the cleaning of the data and approaches 1-3

Cleaning the data

Reading in the data

```
# Import needed packages

import pandas as pd
import csv
import numpy as np
```

```
# Reading in the trips as csv
# Specifying how many rows to read it
initial_trips_df = pd.read_csv("40.trips.csv", sep=",", nrows=1000)
```

Changing the time formats

```
# Change format of travel time
initial_trips_df["trav_time_sep"]=initial_trips_df['trav_time'].astype("str").str.split(":")


# Calculate trav_time in seconds
initial_trips_df['h']=initial_trips_df['trav_time_sep'].str[0]
initial_trips_df['m']=initial_trips_df['trav_time_sep'].str[1]
initial_trips_df['s']=initial_trips_df['trav_time_sep'].str[2]

# Update the old trav_time by the calculated one in seconds
initial_trips_df['trav_time'] = initial_trips_df['h'].astype("int")*3600
    + initial_trips_df['m'].astype('int')*60 + initial_trips_df['s'].astype('int')
```

```
# Change format of departure time
initial_trips_df['dep_time_sep']=initial_trips_df['dep_time'].astype("str").str.split(":")

# Calculate departure time in seconds
initial_trips_df['h']=initial_trips_df['dep_time_sep'].str[0]
initial_trips_df['m']=initial_trips_df['dep_time_sep'].str[1]
initial_trips_df['s']=initial_trips_df['dep_time_sep'].str[2]

# Update the old departure time by the calculated one in seconds
initial_trips_df['dep_time'] = initial_trips_df['h'].astype("int")*3600
    + initial_trips_df['m'].astype('int')*60 + initial_trips_df['s'].astype('int')
```

Dropping the trips where the mode is not a car

```
trips1_df = initial_trips_df[(initial_trips_df['modes']=='car')]
```

Dropping the rows where start and end location are the same

```
trips2_df = trips1_df[(trips1_df['start_x']+trips1_df['start_y'] - trips1_df['end_x'] - trips1_df['end_y'])**2 >1]
```

Taking only the needed columns

```
trips3_df = trips2_df[['person', 'dep_time', 'trav_time', 'wait_time', 'start_x', 'start_y', 'end_x', 'end_y', 'trip_id']]
```

Resetting the index and sorting the data set after the departure time

```
trips4_df = trips3_df
trips4_df = trips4_df.reset_index(drop=True)
trips5_df = trips4_df.sort_values('dep_time')
```

Reducing the number of digits for the 'person' column; creating a data frame containing the unique values of the filtered data set

```
cleaned_trips_df = trips5_df[['person', 'dep_time', 'trav_time', 'wait_time',
                              'start_x', 'start_y', 'end_x', 'end_y', 'trip_id']].copy()
cleaned_trips_df['person'] = cleaned_trips_df['person'] - 201740002800000
cleaned_id = cleaned_trips_df.person.unique()
```

Dropping the agents with inconsistent trip locations

```
# Sorting the data set after the trip id and resetting the index
cleaned_trips_df = cleaned_trips_df.sort_values('trip_id')
cleaned_trips_df = cleaned_trips_df.reset_index(drop=True)


# Iterating over the unique values dataframe of the filtered data
for ID in range (len(cleaned_id)):

    # Adding agent 'ID''s trips to a temporary data frame and resetting the index
    temporary_df = cleaned_trips_df[cleaned_trips_df['person'] == cleaned_id[ID]]
    temporary_df = temporary_df.reset_index(drop=True)

    # Checking, to see the agent does at least two trips (roundtrip)
    # Dropping the agent, if that is not the case
    if len(temporary_df)<= 1:
        cleaned_trips_df = cleaned_trips_df[cleaned_trips_df['person']!=cleaned_id[ID]]

    # Iterating over the temporary data frame
    for trip in range (len(temporary_df) - 1):

        # Checking if the following trip's starting location is the same as the actual trip's ending location
        # Dropping the agent, if that is not the case
        if temporary_df.loc[trip + 1, 'start_x'] != temporary_df.loc[trip, 'end_x']:
            cleaned_trips_df = cleaned_trips_df[cleaned_trips_df['person']!=cleaned_id[ID]]

# Resetting the index
cleaned_trips_df = cleaned_trips_df.reset_index(drop=True)
```

Creating a data frame containing only agents, who return to their initial location

```
# Creating an empty data frame, same characteristics as the filtered data set
All_Trips_df = pd.DataFrame(columns = ['person','dep_time','trav_time',
                                       'wait_time','start_x','start_y','end_x','end_y','trip_id'])

# Sorting the data set after the trip id and resetting the index
cleaned_trips_df = cleaned_trips_df.sort_values('trip_id')
cleaned_trips_df = cleaned_trips_df.reset_index(drop=True)

# Iterating over the unique values dataframe of the filtered data
for x in range (len(cleaned_trips_df.person.unique())):

    # Adding agent 'x''s trips to a temporary data frame and resetting the index
    temporary_df = cleaned_trips_df[cleaned_trips_df['person'] == cleaned_trips_df.person.unique()[x]]
    temporary_df = temporary_df.reset_index(drop=True)

    # Checking, to see the agent returns to his initial position with his last trip
    # Adding the agent to the All-Trips data frame, if that is the case
    if temporary_df.loc[0,'start_x'] == temporary_df.loc[len(temporary_df) - 1, 'end_x']:
        All_Trips_df = All_Trips_df.append(temporary_df)

# Sorting the data set after the trip id and resetting the index
All_Trips_df = All_Trips_df.sort_values('trip_id')
All_Trips_df = All_Trips_df.reset_index(drop=True)

# Creating a data frame containing the unique values of the filtered data set
All_Trips_id = All_Trips_df.person.unique()
```

Generated: a data frame with agents making car trips with consistent locations and returning home after their last day - 'All-trips'.

Preparing the data frames needed for all three approaches

Creating a data frame only containing the first and last trip of all subtours of all the agents - 'Subtours'

```
# Defining a temporary dataframe with the All-trips data, which will be left empty after the for-loop
# Resetting the index
Temporary_df = All_Trips_df.sort_values('trip_id')
Temporary_df = Temporary_df.reset_index(drop=True)

# Creating a data frame to put the first and last trip of a subtour into
# Same characteristics as All-trips
Subtours_df = pd.DataFrame(columns = ['person', 'dep_time', 'trav_time','wait_time','start_x',
                                      'start_y','end_x','end_y','trip_id'])
```

```
# Filling up the Subtours data frame to contain the first and last trip of the subtours of an agent by
# locating the next trip with the initial start location of the agent,
# Dropping the rows of the temporary data frame up until the last trip of the subtour 'x'
for x in range (len(Temporary_df)):

    # Checking, that the temporary data frame is not empty
    if len(Temporary_df)!=0:

        # Filling a Selected data frame with trips done by the same agent and
        # ending at the starting location of the 0th trip of the temporary data frame
        # Resetting the index
        Selected_df = Temporary_df[Temporary_df['end_x'] == Temporary_df.loc[0,'start_x']]
        Selected_df = Selected_df[Selected_df['person'] == Temporary_df.loc[0,'person']]
        Selected_df = Selected_df.reset_index(drop=True)

        # Checking, that the Selected data frame is not empty
        if len(Selected_df)!=0:

            # Locating the index of the first element of Selected
            index = Temporary_df[Temporary_df['trip_id']==Selected_df.loc[0,'trip_id']].index.values.astype(int)
            int_index = index[0] + 1

            # Checking, that the index is not empty
            if len(index)!=0:

                # Appending the first and last trip of the subtour to Subtours
                Subtours_df = Subtours_df.append(Temporary_df.loc[0])
                Subtours_df = Subtours_df.append(Temporary_df.loc[index])

                # Dropping all the trips up until the last trip of the actual subtour
                # Resetting the index
                Temporary_df = Temporary_df.iloc[int_index:]
                Temporary_df = Temporary_df.reset_index(drop=True)

        # There is no trip ending at the start location of the 0th element of the temporary data frame
        else:

            # Dropping the 0th element of the temporary data frame
            Temporary_df = Temporary_df.iloc[1:]
            Temporary_df = Temporary_df.reset_index(drop=True)

# END of for-loop


# Sorting the Subtours after the trip id and resetting the index
Subtours_df = Subtours_df.sort_values('trip_id')
Subtours_df = Subtours_df.reset_index(drop=True)
```

Creating an array containing the unique agent ('person') values of Subtours

```
# Sorting Subtours after the departure time
# (in the iteration the earliest agents will take their own car)
sorted_Subtours_df = Subtours_df.sort_values('dep_time')

# Filling the array with unique 'person' values
Subtours_id = sorted_Subtours_df.person.unique()
```

Creating the 'Subtours-Condensed' data frame with the subtours from 'Subtours' condensed to one row per subtour

```
# creating the 'Subtours-Condensed' data frame with columns: agent, departure time, ending time, x-location, and y-location
Subtours_Condensed_df = pd.DataFrame(columns=['person', 'start','end','loc_x', 'loc_y'])

# Creating a temporary data frame with 'Subtours' data, sorted by the trip id
# This data frame is getting emptied as the rows are copied to 'Subtours-condensed'
# Resetting the index
Temporary_df = Subtours_df.sort_values('trip_id')
Temporary_df = Temporary_df.reset_index(drop=True)

# Iterating over the length of the temporary data frame
for i in range (len(Temporary_df)):

    # Stopping the iteration if the temporary data frame is empty
    if len(Temporary_df) == 0:
        break

    # Generating a 'Selected' data frame containing the trips of the actual agent
    # which end at this subtour's initial starting position
    selected_df = Temporary_df[Temporary_df['person']==Temporary_df.loc[0,'person']]
    selected_df = selected_df[selected_df['end_x']==Temporary_df.loc[0,'start_x']]

    # Given that no suitable last trip of the subtour is found, the agent is dropped from the temporary data frame
    # and the index is reset
    if len(selected_df) == 0:
        Temporary_df = Temporary_df[Temporary_df['person'] != Temporary_df.loc[0,'person']]
        Temporary_df = Temporary_df.reset_index(drop=True)
```

```
    # There is a suitable last trip of the subtour found
    # The index of 'Selected' is reset
    # The information of the first and last trip of the subtour is combined in an array
    # 'Subtours-Condensed' is appended the array
    else:
        selected_df = selected_df.reset_index(drop=True)
        person = selected_df.loc[0, 'person']
        start = Temporary_df.loc[0,'dep_time']
        end = selected_df.loc[0,'dep_time'] + selected_df.loc[0, 'trav_time']
        loc_x = selected_df.loc[0,'end_x']
        loc_y = selected_df.loc[0,'end_y']
        new_row = {'person': person, 'start': start, 'end': end, 'loc_x': loc_x, 'loc_y': loc_y}
        Subtours_Condensed_df = Subtours_Condensed_df.append(new_row, ignore_index = True)

        # Locating the last trip of the subtour in the temporary data frame
        # Dropping all rows up until the index of the last trip
        # Resetting the index of the temporary data frame
        prov_index = Temporary_df[Temporary_df['trip_id']==selected_df.loc[0,'trip_id']].index.values.astype(int)
        index = prov_index[0] + 1
        Temporary_df = Temporary_df.iloc[index: , :]
        Temporary_df = Temporary_df.reset_index(drop=True)


# Sorting the Subtours data frame after the departure time and resetting the index
Subtours_Condensed_df = Subtours_Condensed_df.sort_values('start')
Subtours_Condensed_df = Subtours_Condensed_df.reset_index(drop=True)
```

```
# Approach 1
```

Creating a data frame only containing the first subtours of every agent found in 'Subtours' - 'Single-Subtours'

```
# Defining the empty dataframe where the subtours will be filled in
Single_Subtours_df = pd.DataFrame()

# Iterating to fill 'Single-Subtours' by taking the first two trips corresponding to every unique 'person' value
# Locating the first trip corresponding to unique 'person' value 'i'
# and appending this and the following trip to 'Single-Subtours'
for i in range (len(Subtours_id)):

    prov_index = Subtours_df[Subtours_df['person']==Subtours_id[i]].index.values.astype(int)
    index = prov_index[0]
    Single_Subtours_df =Single_Subtours_df.append(Subtours_df.loc[index])
    Single_Subtours_df =Single_Subtours_df.append(Subtours_df.loc[index+1])

# Sorting 'Single-Subtours' after the trip id and resetting the index
Single_Subtours_df = Single_Subtours_df.sort_values('trip_id')
Single_Subtours_df = Single_Subtours_df.reset_index(drop=True)
```

Filling the 'Single-Subtours-condensed' dictionary to include the first subtours of every agent as only one row

```
# Defining the empty dictionary Single-Subtours-condensed to be filled
Single_Subtours_condensed_dic = {};


# Iterating through the unique 'person' values array to combine the first and last trip of a subtour
# to one row in Single-Subtours-condensed
# Locating the first trip made by agent 'i'
# Arranging the information of the first and last trip of the subtour (index and index+1) into an array
# Adding the array to Single-Subtours-condensed
for i in range(len(Subtours_id)):

    prov_index = Single_Subtours_df[Single_Subtours_df['person']==Subtours_id[i]].index.values.astype(int)
    index = prov_index[0]
    person = Single_Subtours_df.loc[index, 'person']
    start_time = Single_Subtours_df.loc[index,'dep_time']
    end_time = Single_Subtours_df.loc[index+1,'dep_time'] + Single_Subtours_df.loc[index+1,'trav_time']
    veh_location = [[Single_Subtours_df.loc[index,'start_x'], Single_Subtours_df.loc[index,'start_y']]]
    row = np.array([start_time, end_time, veh_location, person], dtype=object)
    Single_Subtours_condensed_dic[Subtours_id[i]] = np.vstack([row])
```

APPROACH 1.1 - one subtour per agent; during rental the vehicle is NOT BLOCKED for other renters

Investigating the number of cars (Needed-cars-1-1) needed to satisfy the first subtours of every agent by letting them rent the cars that have already been used for a subtour (during the rental time, the car IS NOT BLOCKED for other renters)

```
# Defining the needed_cars_1_1_dic dictionary to contain the needed cars for the subtours demand
needed_cars_1_1_dic = {};

# Iterating over all subtours to look for potential rental opportunities
for car, subtour in Single_Subtours_condensed_dic.items():

    # Defining a condition that must be fulfilled for the owner's car to be added to Needed-cars-1-1
    Rental_found = False

    # Iterating over needed_cars_1_1_dic to search for rentals
    # If no rental is found, the owner's car is added to needed_cars_1_1_dic
    # If a rental is found, the rent-schedule is NOT updated (simplification in Approach 1.1)
    for needed_car, needed_schedule in needed_cars_1_1_dic.items():
```

```
        # Simplification of names for the elements of Single-Subtours-condensed and Needed-cars-1-1
        new_start = subtour[0][0]
        new_end = subtour[0][1]
        new_x = subtour[0][2][0][0]
        new_y = subtour[0][2][0][1]

        needed_start = needed_schedule[0][0]
        needed_end = needed_schedule[0][1]
        needed_x = needed_schedule[0][2][0][0]
        needed_y = needed_schedule[0][2][0][1]

        # Calculating the distance from the original starting location to the rented car's position
        # Using the Pythagoras theorem and a detour factor of 1.3 to calculate the distance
        dis_x = new_x - needed_x
        dis_y = new_y - needed_y
        dis_tot2 = dis_x**2 + dis_y**2
        dis_tot = 1.3*np.sqrt(dis_tot2)

        # Calculating the additional travel time by assuming a walking velocity of 5 km/h
        # Factor of 2 to account for picking up and bringing back the car
        add_trav_time = 2*dis_tot*(3600/5000)

        # Updating the ending time of the subtour by adding the additional travel time
        new_end += add_trav_time

        # Checking if the distance to the rental car is in range (500 m, 1,000 m, 1,500 m, 2,000 m)
        # and if the car owner's schedule allows for the rental to happen.
        # If the rental can happen, the search for this subtour can be stopped (break)
        if (dis_tot < 2000) and (new_start > needed_end or new_end < needed_start):
            Rental_found = True
            break
    # END of Needed-cars-1-1 loop


    # No rental opportunity has been found
    # The agnet has to take their own car
    # The agent's car is added to Needed-cars-1-1
    if Rental_found == False:
        needed_cars_1_1_dic[car] = subtour

# END of Single-Subtours-condensed loop
```

APPROACH 1.2 - one subtour per agent; during rental the vehicle is BLOCKED for other renters

Investigating the number of cars (Needed-cars-1-2) needed to satisfy the first subtours of every agent by letting them rent the cars that have already been used for a subtour (during the rental time, the car IS BLOCKED for other renters)

```
# Defining the needed_cars_1_2_dic dictionary to contain the needed cars for the subtours demand
needed_cars_1_2_dic = {};

# Iteration over all subtours to look for potential rental opportunities
for car, subtour in Single_Subtours_condensed_dic.items():

    # Defining the condition that must be fulfilled for the agent's car to be added to the Needed-cars-1-2
    Rental_found = False

    # Iterating over needed_cars_1_2_dic to search for rentals
    # If no rental is found, the owner's car is added to needed_cars_1_2_dic
    # If a rental is found, the rent-schedule IS updated
    for needed_car, needed_schedule in needed_cars_1_2_dic.items():
        # Counter to see if whole schedule of the car owner is compatible with the rental
        available_counter = 0

        # Defining a variable for the number of rows in the schedule matrix of needed_cars_1_2_dic
        needed_trips = len(needed_cars_1_2_dic[needed_car])

        # Iterating through the whole schedule matrix of the 'needed_car' to see if the schedules are compatible
        for N in range(needed_trips):

            # Defining simpler names for the elements of Single-Subtours-condensed and needed_cars_1_2_dic
            new_start = subtour[0][0]
            new_end = subtour[0][1]
            new_x = subtour[0][2][0][0]
            new_y = subtour[0][2][0][1]

            needed_start = needed_schedule[N][0]
            needed_end = needed_schedule[N][1]
            needed_x = needed_schedule[N][2][0][0]
            needed_y = needed_schedule[N][2][0][1]

            # Calculating the distance from the original starting location to the rented car's position
            # Using the Pythagoras theorem and a detour factor of 1.3
            dis_x = new_x - needed_x
            dis_y = new_y - needed_y
            dis_tot2 = dis_x**2 + dis_y**2
            dis_tot = 1.3*np.sqrt(dis_tot2)

            # Calculating the additional travel time by assuming a walking velocity of 5 km/h
            # Factor of 2 to account for picking up and bringing back the car
            add_trav_time = 2*dis_tot*(3600/5000)
```

```
            # Updating the ending time of the subtour by adding the additional travel time
            new_end += add_trav_time

            # Checking if the distance to the rental car is in range (500 m, 1,000 m, 1,500 m, 2,000 m)
            # and if the car owner's schedule allows for the rental to happen.
            # If the distance is in range and the rental happens at other times than trip 'N',
            # the 'available_counter' is increased by one to indicate that these two trips are compatible
            if (dis_tot < 2000) and (new_start > needed_end or new_end < needed_start):
                available_counter += 1

            else:
                break


        # If the counter equals the number of rows of the needed_schedule matrix, the rental can take place.
        # This being the case, the needed_schedule matrix can be updated to include this subtour
        # Setting the rental boolean to 'True' - a rental takes place
        # Increasing the rental counter by one - a rental takes place
        if available_counter == needed_trips:
            needed_cars_1_2_dic[needed_car] = np.vstack([needed_schedule, subtour])
            Rental_found = True
            break

    # END of Needed-cars-1-2 loop


    # No rental opportunity has been found
    # The agent has to take their own car
    # The agent's car is added to Needed-cars-1-2
    if Rental_found == False:
        needed_cars_1_2_dic[car] = subtour


# END of Single-Subtours-condensed loop
```

Approach 2

Filling the All-Subtours-condensed dictionary containing all the subtours with one subtour per row

```
# Defining the empty dictionary All-Subtours-condensed to be filled
All_Subtours_condensed_dic = {};

# Iterating through the unique 'person' values array to combine the first and last trip of a subtour
# to one row in All-Subtours-condensed
for i in range(len(Subtours_id)):

    # Searching for all the trips by agent 'i' and copying them into the 'All-Selected' temporary data frame
    # Resetting the index
    All_selected_df = Subtours_df[Subtours_df['person'] == Subtours_id[i]]
    All_selected_df = All_selected_df.reset_index(drop=True)

    # Iterating over the length of 'All-Selected' to identify the first and last trip of every subtour
    # and adding them to All-Subtours-condensed
    for trip in range (len(All_selected_df)):

        # Giving the 'All-Selected' row simpler names
        person = All_selected_df.loc[trip, 'person']
        start_time = All_selected_df.loc[trip,'start']
        end_time = All_selected_df.loc[trip,'end']

        # Generating an array containing departure time, ending time, starting location and agent
        veh_location = [All_selected_df.loc[trip,'loc_x'], All_selected_df.loc[trip,'loc_y']]
        row = np.array([start_time, end_time, veh_location, person], dtype=object)
```

```
        # Given that the agent's car is already included in All-Subtours-condensed:
        # Add to existing schedule the generated row with the actual subtour
        if Subtours_id[i] in All_Subtours_condensed_dic.keys():
            All_Subtours_condensed_dic[Subtours_id[i]] = np.vstack([All_Subtours_condensed_dic[Subtours_id[i]],row])

        # Given that the agent's car is NOT included in All-Subtours-condensed:
        # Create a new dictionary entry with the new subtour array as a 'value'
        else:
            All_Subtours_condensed_dic[Subtours_id[i]] = np.vstack([row])


        # Dropping the subtour from 'All-Selected'
        All_selected_df = All_selected_df.drop(index=trip)
```

APPROACH 2 - All subtours of every agent; car can only be rented at home

Investigating the number of cars (Needed-cars-2) needed to satisfy all the subtours of every agent by letting them rent the cars that have already been used for a subtour. The car can only be rented at the car owner's home and is blocked during the owner's subtours.

```
# Defining the Needed-cars-2 dictionary to contain the needed cars for the subtours demand
needed_cars_2_dic = {};

# Iteration over all subtours to look for potential rental opportunities
for trips in range (len(Subtours_Condensed_df)):

    # Defining the condition that must be fulfilled for the agent's car to be added to the needed_cars_2_dic
    Rental_found = False

    # Defining simpler names for the elements of Subtours-condensed
    person = Subtours_Condensed_df.loc[trips,'person']
    new_start = Subtours_Condensed_df.loc[trips,'start']
    new_end = Subtours_Condensed_df.loc[trips,'end']
    new_x = Subtours_Condensed_df.loc[trips,'loc_x']
    new_y = Subtours_Condensed_df.loc[trips,'loc_y']


    # Checking, if the agent's car has already been used and is in needed_cars_2_dic
    # If yes, the owner makes the trip with his own car - a 'rental' has been found
    if Subtours_Condensed_df.loc[trips,'person'] in needed_cars_2_dic:
        Rental_found = True


    # The agent's car is not in needed_cars_2_dic
    # Checking for rentals
    else:


        # Iterating over needed_cars_2_dic to search for rentals
        # If no rental is found, the owner's car is added to needed_cars_2_dic
        # If a rental is found, the rent-schedule of the car (needed_schedule) is updated
        for needed_car, needed_schedule in needed_cars_2_dic.items():


            # Counter to see if whole schedule of the car owner is compatible with the rental
            available_counter = 0


            # Defining a variable for the number of rows in the schedule matrix of needed_cars_2_dic
            needed_trips = len(needed_cars_2_dic[needed_car])
            # Iterating through the whole schedule matrix of the 'needed_car' to see if the schedules are compatible
            for N in range(needed_trips):

                # Defining simpler names for the elements of needed_cars_2_dic
                needed_start = needed_schedule[N][0]
                needed_end = needed_schedule[N][1]
                needed_x = needed_schedule[N][2][0]
                needed_y = needed_schedule[N][2][1]

                # Calculating the distance from the original starting location to the rented car's position
                # Using the Pythagoras theorem and a detour factor of 1.3
                dis_x = new_x - needed_x
                dis_y = new_y - needed_y
                dis_tot2 = dis_x**2 + dis_y**2
                dis_tot = 1.3*np.sqrt(dis_tot2)

                # Calculating the additional travel time by assuming a walking velocity of 5 km/h
                # Factor of 2 to account for picking up and bringing back the car
                add_trav_time = 2*dis_tot*(3600/5000)

                # Updating the ending time of the subtour by adding the additional travel time
                new_end += add_trav_time

                # Checking if the distance to the rental car is in range (500 m, 1,000 m, 1,500 m, 2,000 m)
                # and if the car owner's schedule allows for the rental to happen.
                # If the distance is in range and the rental happens at other times than trip 'N',
                # the 'available_counter' is increased by one to indicate that these two trips are compatible
                if (dis_tot < 2000) and (new_start > needed_end or new_end < needed_start):
                    available_counter += 1

                else:
                    break
```

```
            # If the counter equals the number of rows of the needed_schedule matrix, the rental can take place.
            # This being the case, the needed_schedule matrix can be updated to include this subtour
            # Setting the rental boolean to 'True' - a rental takes place
            # Increasing the rental counter by one - a rental takes place
            if available_counter == len(needed_cars_2_dic[needed_car]):
                row = np.array([new_start, new_end, [new_x, new_y], person], dtype=object)
                needed_cars_2_dic[needed_car] = np.vstack([needed_schedule, row])
                Rental_found = True
                break


    # END of needed_cars_2_dic loop


    # No rental opportunity has been found
    # The agent has to take their own car
    # The agent's car is added to needed_cars_2_dic
    if Rental_found == False:
        new_loc = np.array([new_x, new_y])
        row = np.array([new_start, new_end, new_loc, person], dtype=object)
        needed_cars_2_dic[person] = All_Subtours_condensed_dic[person]


# END of All-Subtours-condensed loop
```

Approach 3

Filling the All-Trips-Schedule dictionary containing all the trips for each car

```
# Defining the empty dictionary All-Trips-Schedule to be filled
All_Trips_Schedule_dic = {};


# Iteration over all the agents doing trips
for i in range(len(All_Trips_id)):

    # Generating a 'Selected' data frame containing the trips of the actual agent and resetting the index
    selected_df = All_Trips_df[All_Trips_df['person'] == All_Trips_id[i]]
    selected_df = selected_df.reset_index(drop=True)


    # Iteration over all the trips done by the actual agent
    for trip in range (len(selected_df)):

        # Defining simpler names for the elements of the Selected data frame
        person = selected_df.loc[trip, 'person']
        start_time = selected_df.loc[trip,'dep_time']
        end_time = selected_df.loc[trip,'dep_time'] + selected_df.loc[trip, 'trav_time']
        veh_location_start = [selected_df.loc[trip,'start_x'], selected_df.loc[trip,'start_y']]
        veh_location_end = [selected_df.loc[trip,'end_x'], selected_df.loc[trip,'end_y']]

        # Creating an array containing the details for the actual car trip
        row = np.array([start_time, end_time, veh_location_start, veh_location_end, person], dtype=object)

        # Checking if the agent's car is already included in All-Trips-Schedule dictionary
        # If yes, add the actual trip to the existing schedule
        # If not, create a dictionary entry and add the actual trip as schedule
        if All_Trips_id[i] in All_Trips_Schedule_dic.keys():
            All_Trips_Schedule_dic[All_Trips_id[i]] = np.vstack([All_Trips_Schedule_dic[All_Trips_id[i]],row])

        else:
            All_Trips_Schedule_dic[All_Trips_id[i]] = np.vstack([row])
```

APPROACH 3 - Try to fit new subtours into already existing cars' schedules; renting not only at home

Investigating the number of cars (Needed-cars-3) needed to satisfy all trips by filling the already needed cars' schedule with new subtours. Renting is also possible at other locations than at home, i.e. the car is not blocked while the owner has their car parked.

```
# Defining the needed_cars_3_dic dictionary to contain the needed cars for the subtours demand
needed_cars_3_dic = {};

# Iteration over all the subtours of every agent, sorted after the earliest departure time
# Searching for rental opportunities for the subtour
for subtours in range (len(Subtours_Condensed_df)):

    # Defining the condition that must be fulfilled for the agent's car to be added to needed_cars_3_dic
    Rental_found = False

    # Giving the agent of the subtour a simpler name
    subtour_person = Subtours_Condensed_df.loc[subtours, 'person']


    # Iteration over the needed_cars_3_dic to search for possible rentals
    for needed_car, needed_schedule in needed_cars_3_dic.items():

        # Checking, if the agent's car has already been used and is in needed_cars_3_dic
        # If yes, the owner makes the subtour with his own car - a 'rental' has been found
        # Breaking the loop
        if needed_car == subtour_person:
            Rental_found = True
            break

        # The agent's car is not in needed_cars_3_dic
        # Checking for rentals
        else:


            # Defining simpler names for the elements of the actual subtour
            subtour_start = Subtours_Condensed_df.loc[subtours,'start']
            subtour_end = Subtours_Condensed_df.loc[subtours,'end']
            subtour_loc_x = Subtours_Condensed_df.loc[subtours,'loc_x']
            subtour_loc_y = Subtours_Condensed_df.loc[subtours,'loc_y']


            # Generating a schedule_matrix to contain the car's trip schedule
            # Contains departure time, return time, iteration index
            scheduled_trips = len(needed_schedule)
            schedule_matrix = np.zeros((scheduled_trips, 3))

            # Iteration through the needed_schedule of the car to fill the schedule_matrix
            for trip_it in range (scheduled_trips):

                needed_start = needed_schedule[trip_it][0]
                needed_end = needed_schedule[trip_it][1]

                schedule_matrix[trip_it][0] = needed_start
                schedule_matrix[trip_it][1] = needed_end
                schedule_matrix[trip_it][2] = trip_it

            # After filling the schedule_matrix - Sorting the matrix after the earliest departure time
            schedule_matrix = schedule_matrix[schedule_matrix[:, 0].argsort()]


            # CASE 1: The subtour starts and ends before the car is needed
            # Adding the max. walking time to the ending time of the subtour
            # - for walking to the rental location and back
            if (subtour_end + 2880) < schedule_matrix[0][0]:

                # Getting the needed_schedule index of the trip following the subtour rental
                # - thus, getting the rental location of the car
                index = schedule_matrix[0][2]
                index = index.astype(int)
                x_schedule = needed_schedule[index][2][0]
                y_schedule = needed_schedule[index][2][1]

                # Calculating the distance from the original starting location to the rented car's position
                # Using the Pythagoras theorem and a detour factor of 1.3
                dis_x = subtour_loc_x - x_schedule
                dis_y = subtour_loc_y - y_schedule
                dis_tot2 = dis_x**2 + dis_y**2
                dis_tot = 1.3*np.sqrt(dis_tot2)

                # Calculating the additional travel time by assuming a walking velocity of 5 km/h
                # Factor of 2 to account for picking up and bringing back the car
                add_trav_time = 2*dis_tot*(3600/5000)

                # Updating the ending time of the subtour by adding the additional travel time
                subtour_end_2 = subtour_end + add_trav_time

                # Checking if the distance to the rental car is in range (500 m, 1,000 m, 1,500 m, 2,000 m)
                if dis_tot < 2000:
```

```
                # Checking whether the new travel time is smaller than 1.5 times the old travel time
                # If yes, add the subtour to the car's schedule
                # Setting the rental boolean to 'True'
                if (subtour_end - subtour_start)*1.5 > (subtour_end_2 - subtour_start):
                    row = np.array([subtour_start, subtour_end_2, [subtour_loc_x, subtour_loc_y],
                                    [subtour_loc_x, subtour_loc_y], subtour_person], dtype=object)
                    needed_cars_3_dic[needed_car] = np.vstack([needed_schedule, row])
                    Rental_found = True
                    break

        # If-statement to assure the loop is broken completely if a rental has been found
        if Rental_found == True:
            break


        # CASE 2: The subtour starts and ends after the last trip of the car
        if subtour_start > schedule_matrix[scheduled_trips - 1][1]:

            # Getting the needed_schedule index of the previous trip to the subtour rental
            # - thus, getting the rental location of the car
            index = schedule_matrix[scheduled_trips - 1][2]
            index = index.astype(int)
            x_schedule = needed_schedule[index][2][0]
            y_schedule = needed_schedule[index][2][1]

            # Calculating the distance from the original starting location to the rented car's position
            # Using the Pythagoras theorem and a detour factor of 1.3
            dis_x = subtour_loc_x - x_schedule
            dis_y = subtour_loc_y - y_schedule
            dis_tot2 = dis_x**2 + dis_y**2
            dis_tot = 1.3*np.sqrt(dis_tot2)
            # Calculating the additional travel time by assuming a walking velocity of 5 km/h
            # Factor of 2 to account for picking up and bringing back the car
            add_trav_time = 2*dis_tot*(3600/5000)

            # Updating the ending time of the subtour by adding the additional travel time
            subtour_end_2 = subtour_end + add_trav_time

            # Checking if the distance to the rental car is in range (500 m, 1,000 m, 1,500 m, 2,000 m)
            if dis_tot < 2000:

                # Checking whether the new travel time is smaller than 1.5 times the old travel time
                # If yes, add the subtour to the car's schedule
                # Setting the rental boolean to 'True'
                if (subtour_end - subtour_start)*1.5 > (subtour_end_2 - subtour_start):
                    row = np.array([subtour_start, subtour_end_2, [subtour_loc_x, subtour_loc_y],
                                    [subtour_loc_x, subtour_loc_y], subtour_person], dtype=object)
                    needed_cars_3_dic[needed_car] = np.vstack([needed_schedule, row])
                    Rental_found = True
                    break


        # CASE 3: The subtour starts and ends inbetween two trips of the car's schedule
        # Iteration over the schedule_matrix to find the trip preceeding and following the possible rental
        for i in range (scheduled_trips - 1):

            # Checking if the subtour starts after trip 'i' and ends before trip 'i+1' begins
            # - checking the time compatibility of the rental
            # Adding the max. walking time to the ending time of the subtour
            # - for walking to the rental location and back
            if subtour_start > schedule_matrix[i][1] and (subtour_end + 2880) < schedule_matrix[i + 1][0]:
                # Getting the needed_schedule index of the previous trip to the subtour rental
                # - thus, getting the rental location of the car
                index = schedule_matrix[i][2]
                index = index.astype(int)
                x_schedule = needed_schedule[index][2][0]
                y_schedule = needed_schedule[index][2][1]

                # Calculating the distance from the original starting location to the rented car's position
                # Using the Pythagoras theorem and a detour factor of 1.3
                dis_x = subtour_loc_x - x_schedule
                dis_y = subtour_loc_y - y_schedule
                dis_tot2 = dis_x**2 + dis_y**2
                dis_tot = 1.3*np.sqrt(dis_tot2)

                # Calculating the additional travel time by assuming a walking velocity of 5 km/h
                # Factor of 2 to account for picking up and bringing back the car
                add_trav_time = 2*dis_tot*(3600/5000)

                # Updating the ending time of the subtour by adding the additional travel time
                subtour_end_2 = subtour_end + add_trav_time

                # Checking if the distance to the rental car is in range (500 m, 1,000 m, 1,500 m, 2,000 m)
                if dis_tot < 2000:

                    # Checking whether the new travel time is smaller than 1.5 times the old travel time
                    # If yes, add the subtour to the car's schedule
                    # Setting the rental boolean to 'True'
                    if (subtour_end - subtour_start)*1.5 > (subtour_end_2 - subtour_start):
                        row = np.array([subtour_start, subtour_end_2, [subtour_loc_x, subtour_loc_y],
                                        [subtour_loc_x, subtour_loc_y], subtour_person], dtype=object)
                        needed_cars_3_dic[needed_car] = np.vstack([needed_schedule, row])
                        Rental_found = True
                        break
```

```
                    # If-statement to assure the loop is broken completely if a rental has been found
                    if Rental_found == True:
                        break


            # END of schedule_matrix loop


    # END of needed_cars_2_dic loop


    # No rental opportunity has been found
    # The agent has to take their own car
    # The agent's car is added to needed_cars_3_dic
    if Rental_found == False:
        needed_cars_3_dic[subtour_person] = All_Trips_Schedule_dic[subtour_person]

# END of Subtours-condensed loop
```