Ridesharing Using Adaptive Waiting Time

Pooyan Ehsani

A thesis
in
The Department
of
Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science (Quality Systems Engineering)
Concordia University
Montréal, Québec, Canada

August 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Pooyan Ehsani**
Entitled:	**Ridesharing Using Adaptive Waiting Time**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

______________________ Chair
Dr. Nizar Bouguila

______________________ External Examiner
Dr. Daria Terekhov

______________________ Examiner
Dr. Anjali Awasthi

______________________ Supervisor
Dr. Jia Yuan Yu

Approved by ______________________
Dr. Mohammad Mannan, Graduate Program Director

August 7,2020 ______________________

Dr. Mourad Debbabi, Interim Dean
Gina Cody School of Engineering and Computer Science

Abstract

Ridesharing Using Adaptive Waiting Time

Pooyan Ehsani

The culture of sharing by the advances in communication technologies has entered a new era, and ever since, sharing instead of ownership has been sharply increasing in individuals' behaviors. Particularly in transportation, concepts of sharing a ride in either carpooling or ridesharing have been adopting for 70 years. During the past fifteen years, the revolution in communication devices has formed the online version of ridesharing that responds to transportation needs shortly. Ridesharing is considered to be a strategy to mitigate congestion and air pollution by increasing the occupancy rate of vehicles in the road network. An online ridesharing framework is an end-to-end framework that manages the request and matches the passengers to accomplish their rides together.

In this thesis, we studied the online ridesharing problem and proposed an end-to-end framework to handle the passengers. In our end-to-end framework, we design the objective with respect to the passenger's perspective. We assume that all the passengers tend to share their ride to reduce their transportation costs using vehicles. When two passengers get matched to accomplish their ride together, they accept a deviation from their shortest path to make sharing possible. To minimize the information provided by passengers, we define scheduling flexibility using a system-wide fixed flexibility factor ε, which indicates the tolerable increase in travel duration, proportional to the shortest path duration. For a trip, scheduling flexibility is the amount of time that the system has to divide between the detour from the shortest path and the waiting time to find a proper match. To split the scheduling flexibility between the detour and the waiting time, we introduce the concept of adaptive waiting time, which is the key enabler of our framework to provide a quality match for passengers. For a passenger, the optimal waiting time with respect to ε minimizes the expected travel cost. In this work, we use future demand to calculate the expected travel cost.

We carefully design a simulation to observe the ability of our framework to match the passengers. The trip cost and trip duration are approximated using Gradient boosting trees, and we simplify the NYC road network as a grid network. The proposed approach works for 24 hours to handle 356049 ride requests on a rectangle with an area equal to 44 km^2. We analyze several metrics to indicate the quality of the matching process. The simulation results show that by using our approach, 75.2% of the passengers can share their ride by increasing the trip duration for 4.334 minutes on average, and it leads to reducing the total cost by 12% and reducing the total traveled distance by 14.29%.

Acknowledgments

Very truly I tell you, unless a kernel of wheat falls to the ground and dies, it remains only a single seed. But if it dies, it produces many seeds

---

John 12:24

I want to express my sincere appreciation to my supervisor, Professor Jia Yuan Yu, for defining the project and for his support and guidance during the running of this project.

From the bottom of my heart, I wish to express my deepest gratitude to my family and Banafsheh for supporting me and make it easier to resist the hardness of the journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The sharing economy changed over the past thirty years by the expansion of telecommunication technologies. The individuals transformed the culture of sharing to cope with the new era in the sharing economy. In this thesis, we studied an old sharing problem, namely ridesharing, which started during World War II and by the advances in telecommunication and technologies become mature. In this chapter, we investigated the culture of sharing in section 1.1, to understand the intention of the users to participate in a sharing platform. Following that, we indicate the current state of vehicle usage in transportation and explain why we consider vehicles as an underutilized resource in section 1.2. In section 1.3, by looking at the history of ridesharing, we described the ridesharing development from 1942 till today and finally in section 1.4, we describe our ridesharing environment and provide an overview for our ridesharing problem.

1.1 Culture of Sharing

Current sharing economy platforms and services are distinguished for their ability to effectively use underutilized resources in a wide variety of concepts and materials, including homes, tools, vehicles, and knowledge. During the 21st century the widespread usage of the internet and more general information and communications technologies, cause arise of companies and organization which promoted and developed foundations to make sharing possible. These companies took different forms; some are for-profit corporations like Turo, Airbnb, Uber, and Zipcar; others defined themselves as non-profit platforms like Wikipedia, Hoffice, and Bike kitchen. The collaborative commons term used by [75] to denote networked commons like open-source software, and claim these type of commons democratize access to both information and material resources. Advances in digital technologies and online platforms lead to creating new types of commons that would previously have been much more complicated or even practically impossible to organize [13]. In this section, we first analyze the history of sharing economy within humankind and then describe the new form of sharing economy, which becomes available by the advances in technology.

[70] explains that sharing is the most common form of human economic behavior, and it is distinct from and more fundamental than reciprocity. Several anthropological studies have examined the patterns of food sharing in hunter-gatherer societies [54, 44]. On the other hand, [31, 96, 70] argues that there is more than food for sharing, and sharing is not a form of exchange. Consumer research and studies in related disciplines have neglected sharing for a long time, and we briefly explain some of the reasons for this phenomenon. The first reason is that sharing has been treated as a gift exchange or commodity exchange [7]. The second reason is that prior to advances in technologies, sharing happened in intimate societies like families, neighbors, and friends. There is no transactional information available around the sharing, which makes it challenging to analyze the behavior of the participants in an intimate sharing economy [7]. Before the invention of the internet, as described above, sharing appeared between intimate communities in which people could trust each other and had a strong bond to connect socially. Providing a solid definition for sharing is hard as it depends on the context in which sharing occurs. [8] defines sharing as the act and process of distributing what is ours to others, and vice versa.

Sharing before the industrial revolution was limited to natural resources. Peoples who lived in their communities were dependent on localized natural resources and, more importantly, dependent on the continued quality of their resources to survive. Researchers are addressing humankind's use of common natural resources with two different and contradictory analysis. [42] proposed that for any common resources, individuals in the society who make a rational decision according to their self-interest would maximize their benefit. This behavior by individuals will invariably lead to the long-term carrying capacity of the natural resources being exceeded, and subsequently, the community will face a suboptimal outcome, on both collective and individual levels. In game theory, [42] model formalized as prisoner's dilemma [55]. Here we explain the two-player version of the prisoner dilemma by the grazing meadow example.

Let suppose there is a grazing meadow, and two herders want to use the land. For the meadow, there is an upper limit (L) to the number of animals that can graze on the meadow for a season and be well fed. In this example, the cooperate strategy can be thought of as grazing $\frac{L}{2}$ animals for each herder. The defect strategy is any number higher than $\frac{L}{2}$, which the herder may think would maximize his profit. Let suppose the profit of the cooperation strategy is R, when a herder chooses the cooperation strategy, and the second herder chooses the defect strategy, the profit for the first one is S, and for the second one is T. Finally, in the case that both choose the defect strategy, the payoff is P.

	C	D
C	R,R	S,T
D	T,S	P,P

TABLE 1.1: Prisoner's dilemma payoffs

1.1 is the payoff matrix for the two-player prisoner's dilemma and $T > R > P > S$. The problem in prisoner dilemma is that for a rational player (which is the case in game theory) if the column player chooses C, the row player by choosing C would receive R and by choosing D would receive T. If the column player chooses D as his strategy, the row player by choosing C receives S and, by choosing D, receives P. As we can see, D strictly dominates C in this game, and two rational players will receive P by choosing the defect strategy. On the other hand, two irrational players can receive R and the profit of each would be higher than P.

[65] based on extensive empirical studies of natural resource commons situated in localized contexts, including grazing land and fisheries, where participants were dependent on the continued quality of the commons made a different conclusion. [65] observed that, contrary to [42] claim, communities around the world had successfully managed their common resources for decades and more broadly even for centuries. These communities developed proper norms and rules for protecting and using their collective resources. Considering sharing as likely the oldest type of consumption among humankind, and with the rise of technology in the 21st century, it becomes critical to most of the recent consumption phenomena like the internet. Still, the characteristics of digital resources differ from natural resources in two things. First, natural resources are limited and rival; however, we can consider digital resources as unlimited and non-rival. Second, the digital resources concentration is in cyberspace and urban areas, while the natural resources usually exist in

nature [13]. Moreover, advances in digital platforms have also enabled coordination of temporary physical commons; for example, privately owned resources like vehicles, parking lot, and houses temporarily transformed into commons, and other people can use them. [13] has compared three different contemporary sharing commons, including Wikipedia, Bike Kitchen, and Hoffice, concluding that contemporary commons, situated in a globalized, urbanized, and digitalized context, differ from traditional natural resource commons in several aspects. These aspects include the nature of the shared resource, what is rare, barriers to entry and exit the sharing system, reliance on the resource, and how to govern goods.

During the 21st century, digital revolution advancements led to a new form of sharing. In 2019, more than 50% of the human population had access to the internet, and 60% of peoples in advanced economies were using a smartphone. Smartphones are integrated with a digital camera, internet access, a large pool of applications, and a user-friendly interface. Peoples are now massively spending their time in the digital world, forming a new type of social identity, which is called internet identity. All of these developments lead to the creation of a new type of sharing, briefly described here.

The sharing economy in the urbanized area relies on the internet to connect people, and [82] divides it into four broad categories: sharing of productive assets, recirculation of goods, increased utilization of durable assets, and exchange of services. Sharing of productive assets focused on enabling participants to be productive by providing space or assets. For instance, by turning the host home into a shared office space where people can gather and work together, Hoffice provides the space to be productive. Recirculation of goods companies likes eBay and Craiglist creates a proliferation of unwanted items. The main contribution of these systems is reducing the risk of transacting with strangers by providing the reputation of a seller to the buyer and making sharing a part of the mainstream consumer experience. The third type of sharing platform facilitates more intensive usage of durable goods and assets. In transportation, these platforms works in different areas include car rental sites (Turo), ridesharing (Zimride), ride-hailing services (Uber, Lyft, Via), and bike-sharing (Mobike, Bixi). The fourth type formed after the 1980's recession, and the aim was to provide job opportunities for the unemployed. Timebanks sites developed to provide the possibility of trading services based on time spent and remained marginalized in sharing economy.

Sharing platform defined based on their aspects that shape the platforms' business models, logic, and dynamic of the environment where participants interact with each other [82]. Market orientation and market structure have shaped both the operation and long-term impact of sharing platforms and among essential aspects. The market orientation can be either for-profit or non-profit, and the market structure can be either peer-to-peer or business-to-peer. The for-profit platform strategy is to push for revenue and asset maximization. Namely, Uber and Airbnb, two successful models of the for-profit sharing platform, are valued billion dollars. By contrast, many non-profit organizations like Wikipedia, Seed Bank, and Food Swaps do not seek revenue maximization, and instead, aim to serve needs on a global scale like Wikipedia or serve participants locally like Food Swaps. The market structure also has a significant impact on the manner the sharing economy works on a specific platform. The peer-to-peer platform earns money by the commission on exchanges; namely, both Turo and Airbnb, the peer-to-peer platforms, charge the consumer for service fees and also take a portion of the provider. In other words, they take a commission from both sides of the market. So the revenue growth of the peer-to-peer platform is directly linked to the number of trades in their market. On the other hand, the business-to-peer platform aim is to maximize revenue per transaction.

Here our aim is to analyze the components of the sharing economy, and it is essential to define some terms related to it—these terms include providers, consumers, and participants. Providers are offering services or products, while consumers ask for these services or products and together form the participants. As in Airbnb, hosts are providers, and guests are consumers. If we consider Uber, providers are the drivers that serve a rider (consumer) request for transportation. All the sharing platforms exist in a sharing market. One could compare the creation of the sharing market to the creation of the universe from BigBang. The platform provides the necessary context for expanding the market as long as it can sustain itself. The dynamic and the environment of the sharing economy are described here on the perspective of the authors. The sharing economy includes a sharing platform that designs the mechanism of the market and guarantees the operation of itself over time. The other side of the sharing economy is the sharing market, which includes a set of users that generate the dynamic of the market by providing their supply and demand. The main objective of the sharing platform then is to provide users with a designed environment. The environment, which defines the rules of the market, also provides tools that users use in their interactions. All the users are individuals who could have supply or demand and interact with each other to trade at time t.

The sharing platform's goal is to ease the sharing between strangers. It could be a Wikipedia page that shares the knowledge to everyone who accesses the internet and can understand the content. In the case of Uber, the goal is to match the drivers who are joining the market to earn money with riders who join the market to meet their transportation needs. In Uber, case participants can rate each other based on their experience during the trip and deploy a secure payment method to ensure the safe interaction between drivers and riders. The sharing platform creates the space for a sharing market where individuals join the market and contribute to keeping the sharing economy and platform alive. The two sides of the market (providers and consumers) play an essential role in sharing economy. The expansion of the market provides individuals a variety of choices, which increases the quality of trades. After joining a sharing market, each individual has a personalized experience based on their needs, and it is for them to decide to remain in the market or leave. The critical mass is a set of market states in which a significant part of individuals are satisfied with the quality of the trades while tending to remain in the market. The critical mass work as social proof to make a sharing platform acceptable [28].

Two important terms need to be addressed here. The first term is the supply and demand pool, which defined as the supply and demand of all participants at time instance t. As the supply and demand pool increase in size, the sharing platform gets a higher chance of satisfying the participants concerning their needs. The other term is matching the participants based on their needs. The matching process is to find the proper supplier for each demand which becomes available to the sharing market. Sharing platforms have different approaches for matching processes according to their characteristic. In the case of Uber, when a demand becomes available (a participant sends the request for a ride), close suppliers in terms of distance get a notification on their phones and have a limited time to decide to accept or reject the demand. As soon as a driver accepts a rider's request, the two participants match; the supplier dispatches to fulfill the rider's need. In the case of Airbnb, the supplier lists their available properties, and those looking for a place can search the list to find and further book their desired rental. In the first example, suppliers (drivers) decide to accept a passenger or not, based on the system notification, and in the second model, the participants with demand choose between the available places based on their personalized needs.

As we describe in this section, sharing is one of the most primitive forms of interaction in humankind, which evolves, based on the form of community. In the context of a communal sharing relationship, people treat the material as everyday objects in which every part of the commune can use that object [31]. Evidence shows that hunting-gathering societies, share the meat of animals across the whole community [19]. By the evolution of human society, sharing takes a different form and remains common between the peoples who trust each other within small local communities. Urbanization is a new phenomenon in human civilization, and before 19 century, most of the human population lived in rural and low-density societies [41]. During this long duration of humankind's history due to living in low-density societies, peoples have grown an intimate bond with each other and act together. Sharing has a very different form and has played a fundamental role in the survival of the community. Domination of urbanization and the high-density societies confined sharing to people who had trust in each other, e.g., kinship relations and neighbors. After the internet revolution in the late 20th century, sharing again started to rise and drew much attraction for researchers, societies, entrepreneurs, and investors. These days people can rely on each other to share their equipment, rent their cars and places based on the trust systems developed, and also secure payment methods deployed. In the next section, we provide information about the vehicles as an underutilized resource in densely urbanized areas.

1.2 Vehicles as Underutilized Resource

More than 4 billion people live in urban areas, and for the first time in 2007, the urban population exceeded the rural population [76]. In North America, around 80% of the population lives in an urban area. More than 8.5 million people live in New York City, and the density of the population is $\frac{10{,}715}{km^2}$. Handling massive populations' transportation needs requires a network of complex infrastructure for transportation, including road networks, subways, and buses. It enables people to choose their transportation mode based on their needs, including flexibility, fare, and comfort. In New York City, where millions of people use public transportation; still, vehicles are regarded as one of the dominant modes of transportation.

Ride-hailing companies like Uber, Lyft, Juno, and Via have served more than one million requests per day in 2019 in New York City. For the first time in New York City, ride-hailing apps have surpassed the taxicabs, which used to dominate the door-to-door transportation market for decades. Ride-hailing companies in 2017 introduced their sharing services, in which riders could request to share their ride with a stranger to reduce their trip cost. Uber alone has received 175000 (24%) requests for shared trips, which occurred between two passengers daily in NYC. These new technologies have created new opportunities to match people with similar itineraries and time schedules, allowing them to share their rides dynamically and on short notice [1]. These advances can provide society with new opportunities for reducing traffic congestion, fuel consumption, emission, and pollution, which subsequently increases the quality of life in the urbanized areas. The cost of congestion in the United States is relatively high and tends to get worse over time. In 2012, the cost of congestion was roughly 121 billion dollars [84], and in 2017, it increased by 19% and achieved 179 billion dollars [83]. The mentioned cost can be divided into the following categories:

1. Travel delay
    - In 2012, 7.7 billion hours lost to sitting in traffic.
    - The number increased by 14% in 2017 and reached 8.8 billion hours.
2. Wasted fuel
    - In 2012, 3.2 billion gallons of fuel burned in congestion and wasted.
    - By 2017, this number increased by 3% and reached 3.3 billion gallons of wasted fuel.

Today, passengers in urban areas have different modes of transportation to go from their origins to their destinations. Passengers consider a set of criteria to choose their mode of transportation or a mixture of them, including travel cost, travel time, flexibility in schedule, convenience (such as pickup and dropoff locations, privacy, and personalized space), reliability and security. Here we explain the mentioned criteria in different modes [37].

We start by considering public transportation, which provides transportation options with a fixed spatial route and schedule. Public transportation charged a small fee to passengers to use their services, and it comes with a little convenience. Travel time in public transportation is higher than that of alternative modes due to walking distance to the station and stops during the trip. Taxicabs is another mode of transportation that comes at a higher cost due to traveling solo. Taxi cabs provide door to door transportation, and passengers have flexibility in schedule. The travel time using taxi cabs is relatively lower than public transportation, and passengers have personal space and privacy. Ride-hailing apps are the most recent transportation mode and introduced in 2009. The cost of ride-hailing apps is relatively equal to taxi cabs and considered as the most convenient transportation mode, travelers, have door-to-door transit and personalized space, and as a result of the ranking system, people had a higher sense of security and privacy. Ride-hailing apps are the most reliable transportation mode, and the pickup and travel time are estimated accurately. Ridesharing is between public transportation and ride-hailing apps. The travel cost is less than the ride-hailing option due to sharing the vehicle instead of traveling alone, and it is also standing between public transportation and ride-hailing in convenience and travel time. The last transportation mode is private vehicles, which are not available to all of the passengers. The cost is high, and travelers need to handle the parking space but are the most convenient form of transportation by providing personal space, door-to-door transportation, and flexible schedule.

Ridesharing refers to a mode of transportation in which passengers share a vehicle for their trip and split the travel costs. Ridesharing is a combination of traveling solo and public transportation. Passengers have the flexibility and speed of private cars and can furthermore reduce their travel costs by sharing their ride but at the price of convenience. The ridesharing concept has several advantages for individuals, society, and the environment, as we provide in the following list.

1. Individual interest
    - Reduce travel costs by sharing the travel expenses between involved passengers.
    - Reduce travel time and congestion by efficiently using empty seats in vehicles.
    - Increases the possibility of socializing between the passengers who are involved.
2. Social benefit
    - Mitigate traffic congestion by reducing the number of cars in the road network.
    - Conserve fuel by sharing the travel path between passengers who travel solo.
    - Reduce air pollution by a mixture of faster trips and reducing the overall traveled distance by vehicles.

3. Environmental benefit

    - Reduces greenhouse gas emissions in urbanized areas.
    - Increases the quality of soil, water, and air by reducing pollution.

Transportation is responsible for approximately 30% of greenhouse gas emissions in North America, and the private vehicles had more than half of this emission [101]. In the United States, private car occupancy rates are low, and the average for commute trips is 1.18, and leisure trips average is 1.82 [78]. In a large city at peak hours, the city network faces traffic congestion resulted from the high demand for vehicle transportation and relatively low occupancy rates. This misbehavior in transportation has billions of dollars cost and billions of wasted hours for society and individuals. Furthermore, vehicle transportation dominates other transportation modes in producing greenhouse gases [46]. The massive amount of pollution that is produced by vehicles gives rise to health issues related to pollution, which is a severe problem in many of the densely populated regions worldwide [16].

Ridesharing can be a practical approach to reduce congestion by increasing the occupancy rate of vehicles. Ridesharing is not a new idea, and World War II was the genesis of the concept. However, as we mentioned earlier, the massive pool of the users (critical mass) is the main factor in forming an affordable and efficient model for any sharing market and ridesharing also is not excluded from this fact. Technological advances in both hardware and software are the key enabler to reach this massive pool to form a proper and convenient ridesharing system, and currently, companies like Uber and Lyft have this massive pool of users. These successful implementations of the ridesharing have changed the trends in how we tackle the ridesharing problem. Here we begin by briefly describing the history of ridesharing and the current state of ridesharing.

1.3 Development of Ridesharing

The ridesharing concept started in the United States during the world war II, between 1942-1945, and the main reason for sharing transportation was to reduce the consumption of resources and to conserve goods for the war [20]. When tensions between the United States and Japan escalated, Japan cut off 97 percent of US rubber supply, and the US government faced a crisis for supplying goods to the war. The government began to change

American citizens' transportation behavior to reduce the consumption of limited resources [36]. During this period, a wide range of communities, including factories, companies, and churches, were responsible for forming a bulletin board to increase the vehicles' occupancy rate to save goods. The next phase of promoting the ridesharing started in the late 1960s and early 1970s as a response to the energy crisis and the Arab oil embargo of 1973 [20]. During this period, different strategies were introduced to facilitate ridesharing. these strategies included: vanpooling, employer-sponsored commuter ride-matching and HOV lanes [20]. Throughout the energy crisis, the US government decided to reduce petroleum consumption, and different actions were taken to decrease consumption. High Occupancy Vehicle (HOV) lanes were proposed to encourage riders to use their empty seats. In employer-sponsored commuter ride-matching, major employers collected their employee's transportation information and matched those who were neighbors.

Through this stage for the first time, ridesharing was considered a tool to mitigate air quality problems [49]. The environmental and social benefits of ridesharing were addressed during this time due to government-funded rideshare initiatives. A study found that the number of commuters raised by 29400, and the vehicle-miles traveled among 197000 employees reduced 23% [69]. [68] shown empty seats in cars are underutilized resources and to efficiently occupy them results in the following advantages: by reducing congestion, overall travel speed increases, and the overall fuel consumption will decrease. After all these hopes and becoming clear the merits that sharing rides could bring along, during the 1980s due to low oil prices and steady economic growth, ridesharing was marginalized till the 21st century. In the early 2000s, by advances in internet services, initial online ride-matching services were introduced. The carpool services developed during this period were static and required prearrangement. These improvements made it easier to find ride-matches in a more massive online database; however, due to the systems' inflexibilities, passengers preferred to use their own vehicle to commute. During the early stages of ridesharing, owing to psychological barriers, riders ended up deciding not to share their ride. However, technological advances weakened such psychological barriers and increased the success rate of ridesharing [39]. As an example, [32] based on an empirical study reported that people who own a profile on social networking websites have higher risk-taking attitudes and are subsequently less affected by trust issues for sharing their ride with a stranger.

During the past twenty years, several initiatives and matching platforms have been proposed to ease the user experience and facilitate the interaction between the demand and supply sides of the market, which consists of riders and drivers. Most of the ridesharing platforms work on the static form of the process, meaning that users need to plan their rides in prior. In other words, users should know in advance all the details related to their trips, such as the time and location of departure and arrival. In this manner, ridesharing experience is similar to booking a flight. In essence, the user chooses their flight search engine like Flighthub and provides the information related to their interest, including the pickup and dropoff location and the departure time, and then picks among different proposals that satisfy their constraints. The mentioned model works with one-way long trips like the carpooling concept between Montreal and Toronto. However, for a short trip frequently occurring for individuals in an urban area, it is disturbing to a user to provide all the required information and looking for available options. Urban transportation is a massive and dynamic system, and the ridesharing platform needs a smarter system to handle the requests within a few minutes. It further needs to find matches concerning the current state of the system and serve the passengers' demand with a minimum amount of information provided by the user. Few initiatives satisfy the suggested criteria that are available worldwide, and we can mention UberPool and Lyft Line as the most successful examples for ridesharing.

[85] evaluates prices and service levels for Lyft, Lyft Line, UberX, UberPool, and Chicago Transit Authority (CTA) services in Chicago, the third-largest city in the United States. [85] used a stratified sample of 3,075 fares and travel time estimates and evaluated the costs, time, predictability, and convenience associated with each trip made, the stratified sampling was used to take the sample from different spatial characteristics. This sampling set includes trips taken within the city's central business district, neighborhoods, and inter-neighborhoods. Notably, inter-neighborhood traveling tends to be more difficult using public transit. They reported that the average CTA fare is $2.69, and for the Lyft and UberX, the average fare is $18.13 and $17.90 respectively, while for the Lyft Line and UberPool the average fare was $14.04 and $9.33. Their observations indicated that the saving from using Lyft Line comparing to Lyft is 22.6%, whereas passengers using UberPool saved 47.9% over UberX. They also notice that the highest saving (25.1%) from Lyft Line is occurred in downtown, while the most significant saving (49.9%) occurred in the outer downtown zone in Uber case [85].

The travel time is the other important factor addressed in their studies, and based on their observations, UberPool and Lyft Line trips are orderly four minutes (7.6%) and 10 minutes (21.5%) faster than public transit trips. On the other hand, UberX and Lyft are 19 and 22 minutes (38.2% and 45.8%) faster than public transit, respectively. The travel time varies from neighborhood to neighborhood, for example, in the downtown area, UberPool is four minutes (9.9%) slower than public transit, and Lyft Line is two minutes (4.2%) faster than public transit. However, in the neighborhood zone, Uber and Lyft offer much faster trips compared to public transit, which is partially due to longer walking distance to bus and train stops. For the neighborhood zone, UberX and Lyft saving an average of 24 and 20 minutes (46.1% and 39.1%), respectively, and for the sharing options, which are UberPool and Lyft Line, the saving times are 15 and 4 minutes (28.8% and 7.8%), respectively. Based on their studies, transportation network companies (TNCs) are a costly alternative for public transit, but they can reduce the travel time up to 50%. The other important side of their studies is that both the spatial and temporal aspects of trips have a significant impact on the mentioned results.

In 2017 for the first time, ride-hailing apps surpassed regular taxis in New York City with 159.9 million rides. The massive request on the ride-hailing app has two facets. First, handling this amount of data is a challenging computational task and requires an efficient algorithm that would be able to run in short notice and provide efficient results for the matching process. Second, the massive pool of requests makes it possible to reduce the travel cost as much as possible due to a large number of nominated supplies for demands.

1.4 Our Ridesharing Approach

Here we describe our ridesharing platform and how the participants interact with it to fulfill their transportation needs. The ridesharing, in general, has three components, riders, drivers, and the decision-maker. In chapter 2, we provide evidence that shows how the human behavior of drivers leads to a nonoptimal output of the ridesharing system. As [3], in this work, we consider a fleet of autonomous vehicles that the decision-maker controls their movement for better performance. The Uber-problem in [26], can be used as a problem which can address the dispatching, serving, and rebalancing the vehicles in ridesharing platform. To explain our ridesharing platform, we need to describe the different parts of the system in which participants adapted to fulfill their transportation needs. In this work, we consider a central decision-maker with a fleet of autonomous vehicles who want to serve the transportation requests that come over time online. The decision-maker is responsible for matching passengers to each other and afterward dispatch an autonomous vehicle to serve the matched passengers together. We assume that the central decision-maker has the optimal policy for vehicle movement to serve requests and minimize their movement. The policy includes a taxi distribution policy during their cruising time and policy to serve a set of requests. In this work, we assume the decision-maker has a large fleet, which can serve the passenger in short notice without wasting time picking up the passenger.

In this work, we concentrate on the riders' side of the ridesharing. The participants (riders) are the critical component of any sharing market and could increase the system's performance by shifting the system dynamic towards "critical mass". In our setting, the participants are individuals that used sharing platform to meet their transportation need. Each participant has a user profile, and at time t sends a request to the system for traveling from location p to location d. We consider that all the participants are willing to share their rides to accomplish a trip together. From the passenger's perspective, sharing rides and a trip path has the following benefits:

1. Reduce travel costs.
2. Increase the chance of socializing with a diverse range of people.
3. Reduce the environmental loss related to transportation.

We assume that all the passengers tend to share their trip; in other words, passengers share their path with other riders who have similar itineraries and time schedules to join him/her to accomplish a trip together. So based on the idea of sharing trips, when a participant sends a request to travel from location p to location d at time t, supplies a trip to the sharing market, and the demand is the proper rides in the market who can share their rides with them. From the passenger perspective, one of the main reasons for joining a ridesharing market is to reduce the travel cost. In this work, we aim to minimize the transportation cost for each travel request. We consider an online version of ridesharing where the upcoming requests are unknown, and we do not consider en-route matching, and passengers need to be matched before a vehicle dispatch to serve their request. Based on our ridesharing settings, we defined an adaptive waiting time that tries to minimize the expected travel cost for each request. When a travel request from location p to d comes at time t; we use the predicted future demand to calculate the expected travel cost in the near future and select the one that minimizes the expected travel cost.

The literature approaches to deal with time constraints in ridesharing are differing, and here we summarize their formulation and then provide our approach to deal with time constraints. Time constraints formulation can follow into the following categories:

1. Time windows defined by the passenger:
    - [72, 27, 5, 2] used time windows.
    - Riders and drivers define their time windows for both pickup and dropoff.
    - The goal is to serve passengers in their time windows.
    - Users have an inconvenient experience to define the time windows.
2. Fixed waiting time and detour:
    - [90, 3] used fixed time and detour.
    - The waiting time and detour from the shortest path fixed for all the trips.
    - Constant scheduling flexibility is a common technique in practice, and ridesharing platforms like Uberpool use it.
3. Detour proportional to the shortest path:
    - [4] used this technique.
    - In this approach, passengers specify the maximum excess for travel time they are willing to accept.

In this work, we defined system-wide fixed flexibility factor ε, and for a trip with the shortest path equal to t minutes, the system must serve the request at most $(1+\varepsilon)t$. This approach leads to adaptive waiting time based on the upcoming requests, and as we show in the experiments, it achieves competitive results. The other advantage of this approach is to minimize the information users need to provide about their trip schedule. To illustrate our approach for handling time constraints in the ridesharing problem, suppose a passenger decided to travel from his home to his workplace with direct travel time equal to 30 minutes and $\varepsilon = 0.4$. For this passenger, the overall travel time, including waiting time and actual travel time, equals to 42 minutes, which can be divided among the detour from the shortest path and waiting time. It is important to note that in figure 1.1, the scheduling flexibility is

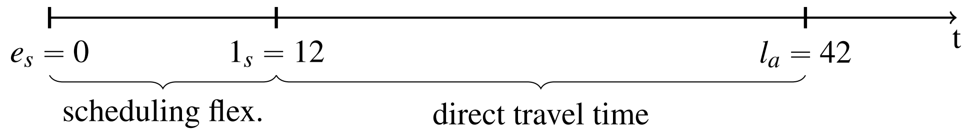


FIGURE 1.1: Scheduling flexibility for a rider request in our approach

divided between the waiting time and detour from the shortest path, caused by participating in ridesharing. To deal with ridesharing's online nature, we define a waiting time for each passenger who sends a ride request. When the waiting time for at least one passenger elapses, the matching procedure runs to match the leaving passengers to available rides. As mentioned earlier, the waiting time aims to use the upcoming request to minimize the cost of the trip for a passenger. When passengers are matched to each other, a vehicle dispatches to serve their trips based on the order that minimizes their cost. To the best of our knowledge, this thesis is the first work to handle vehicles and riders separately. All the previous works in ridesharing, assign the trips to vehicles, instead, we match passengers to each other, and then the vehicles optimize their routes to serve matched passengers. In conclusion, we define the *Serversharing*-problem, a variation of *Uber*-problem, that each server serves a bag of requests instead of one. Dividing the matching and serving request had the computational advantage by reducing the subproblems size.

In this framework, the passengers' shared information is less than the current ride-hailing platforms. Currently, the ride-hailing platforms share the travel information of a rider with the driver, and also, the platform itself has access to the data. As a consequence of eliminating the driver from the proposed framework, the riders' data is only shared with the ridesharing platform. These days, different platforms we massively use collect the user's data to provide a personalized experience for them. The individual transportation data is part of their privacy, and system providers can use it to push for their revenue maximization. This phenomenon can be considered as the dark side of the data-driven systems. During the past five years, consumers become aware of the importance of their data and more concerned about the usage of their data in companies; it leads to an increase in security and regulation standards to protect the individuals' privacy.

The remaining of this thesis as follows: In chapter 2, we excessively review the literature in ridesharing and a wide range of problems related to ridesharing, and we also review the latest advances in machine learning which could affect the ridesharing. We provide the formal definition for our problem and discuss our approach to develop a ridesharing system in chapter 3. In chapter 4, two essential functions are estimated using a data-driven function approximation approach to make the ridesharing simulation possible. The simulation environment and the ridesharing results are provided in chapter 5, and in chapter 6, we provide the conclusion and discuss the future direction to investigate by researchers. Before starting the literature review in the next chapter, we briefly provide the contribution of this thesis:

1. We introduce adaptive waiting time in order to minimize passenger expected travel costs.

2. Reduce the ridesharing problem to passenger matching and *Serversharing*-problem.

3. Study the effect of scheduling flexibility on the performance of ridesharing using extensive experiments on NYC yellow-cab data.

Chapter 2

Literature Review in Ridesharing and Related Topics

The concept of sharing a ride is a vast concept and takes different mathematical formulations based on the setting and problem definition. In this chapter, we have studied the literature for three main problems related to ridesharing. After reviewing related work to ridesharing, we review the related concept in the application of machine learning that affects transportation and, more specifically, ridesharing. The problems which we address here are in the following list:

- Pickup and delivery problem (PDP)
- Ridesharing problem
- Mobility on Demand (MoD)

2.1 Pickup and Delivery Problem

The pickup and delivery problem form an important family of routing problems for transporting peoples and goods from their origin to destination. We reviewed the pickup and delivery problem based on [79, 6]. Next, we study a variation of the pickup and delivery problem called the Dial-a-Ride problem concentrating on passenger transportation concerning [23, 47, 24] with focus on literature review and recent developments, respectively. On the Dial-a-Ride problem, as a result of dealing with passengers, a set of constraints were added to the problem, which would control the passengers' quality of the service, making the problem more complicated than the general case.

In the pickup and delivery problem, a fleet of vehicles with size m has to serve a set of transportation requests. Each request has a pickup and dropoff location, and vehicles move on a metric space to serve the requests [79]. The problem's goal is to construct a set of routes for vehicles to satisfy requests under a set of constraints and an objective function. The objective function can be specified based on the problem, but the most general objective function is the total distance traveled by vehicles. There are different aspects of the pickup and delivery problem which define the characteristics of the problem. A fundamental aspect is how requests become available and have two variations, static and dynamic.

In the static case, all requests are available when the routes for vehicles have to be constructed. In a dynamic scenario, part of the requests is known at the beginning, and the rest becomes available over time. In practice, most of the work tries to solve the dynamic case by reducing it to a sequence of static cases [79]. It is important to note that the static problem's results can be considered an upper bound for the dynamic problem. The dynamic case has more challenges than the static case for facing uncertainty in decision making, and also routes need to be reoptimized based on the requests that come over time in short notice.

Time constraints are a vital characteristic of pickup and delivery problems and become an inner part of models for the problem. Considering the time constraints makes the problem more complicated. In a case that there are no time constraints, finding a valid solution for the problem is trivial: arbitrary assign the problem request to vehicles and each request served separately. In the presence of time constraints, finding a feasible solution for the problem becomes $\mathcal{NP}$-hard [79]. On the other hand, time constraints reduce the solution space, and it may be beneficial for the optimization method. The most common form of time constraints for the pickup and delivery problem is the time window. In the time windows representation of time constraints, for each transportation request i, the user defines a time window $[e_i, l_i]$, which denotes the earliest pickup and latest dropoff, respectively. Each request must be picked and dropped in the time windows. The time constraints are related to the quality of transportation, and in the dial-a-ride problem, time constraints gain much value for dealing with passenger transportation and the necessity of convenience for the passenger.

Objective function defines the goal of the model, and a wide variety of them are defined in the literature for pickup and delivery problems [79, 6]. Here we provide the list of the most used objective function for the static case and afterward discuss the dynamic case's objective function.

1. Minimize duration

   - The total duration of a set of routes is the total time that vehicles need to execute their routes.
   - Route duration includes travel times, waiting times, loading, and unloading times.
   - Minimize duration aim is to minimize the total duration of the execution process.

2. Minimize completion time

   - The completion time is the time where the last request served.
   - If all the vehicles start their trip together and at time zero, then the completion time is equal to the maximum route duration.

3. Minimize travel time

   - The travel time is a subset of travel duration, and the goal is to minimize the total time spent on actual traveling between locations.

4. Minimize passenger inconvenience

   - This objective is mostly used when we deal with passenger transportation.
   - Measure in terms of pickup deviation, dropoff deviation, and excess ride time.
   - In demand-responsive situations where passengers request immediate service, the difference between the request time and pickup time is also an essential factor in passenger inconvenience.

5. Minimize the number of vehicles

   - Vehicles and drivers are the most expensive parts in a dial-a-ride system.
   - Minimizing the number of vehicles to serve all requests is part of the main objective.

The dial-a-ride problem (DARP) as a variation of pickup and delivery problem focuses on passenger transportation. The dial-a-ride problem consists of designing vehicle routes and schedules for *n* users who send pickup and delivery requests between origins and destinations, and transport supplied by a fleet of *m* identical vehicles based at the same location called depot. The aim is to plan a set of routes for each vehicle to minimize the cost, serving as many requests as possible, under a set of constraints. The most common example of dial-a-ride-problem is the door-to-door transportation of elderly or disabled people. The main difference between DARP and PDP is the human presence in DARP. In human transportation, user inconvenience must be reduced and balanced against minimizing operating costs [23]. As the PDP, DARP services may operate according to a *static* and or *dynamic* mode.

The simplest case of the DARP is the case that all of the users are served by a single vehicle, known as the single-vehicle DARP. The static case of the single-vehicle DARP is formulated by [71] and solved as a dynamic program in which the objective function is the minimization of the weighted sum of route completion and passenger dissatisfaction. In [72], authors have considered a user-specified time windows on departure and arrival times. Furthermore, they have provided an approach for finding the exact solution with $\mathcal{O}(n^2 3^n)$ complexity, and as a result of the exponential procedure, the largest instance solved by this approach contains nine users. Later, [27] reformulated the static single-vehicle DARP as an integer programming with considering user-specified time windows, vehicles capacity, and precedence constraints. They proposed a dynamic programming approach to find the exact solution, and their approach was to find optimal solutions for up to 40 users.

The more complicated case of the DARP is the multi-vehicle DARP, which considers multiple vehicles instead of the single vehicle. [52] first introduced the problem and provided a heuristics for solving it, their model considers a maximum ride time as a linear function of direct ride time, and it uses a non-linear objective function to assess the quality of the solution from the passenger perspective. The heuristic selects users in order of the earliest feasible pickup time and gradually inserts them into vehicle routes based on the least possible increase of objective function. The algorithm is tested on a real data set with 2617 users and 28 vehicles. For assigning the passengers to a vehicle, one of the commonly used techniques is to defining clusters of users, which is served by a single-vehicle prior to the routing phase [23]. [12] for solving the multi-vehicle DARP has exploited the clustering idea, and after constructing clusters, a single-vehicle DARP approach is used to construct the routes. [29] improved this two-phase approach by creating ”mini-clusters” of users and combining them to form a feasible vehicle route, using a column generation technique. They solved instances derived from real-life data from three Canadian cities, including Montreal, Sherbrooke, and Toronto, with up to 200 instances.

[22] defines the following multi-vehicle static DARP problem: users specify a window on the arrival time of their outbound trip and on the departure time of the inbound trip, maximum ride time associated with each request using a maximum deviation factor from ride time of the direct trip. They proposed a heuristic by applying tabu search. The search algorithm iteratively removes a request and reinserts it into another route. The algorithm tested on randomly generated instances with $24 < m < 144$ and on six data sets (n=200 and 295). In [74], the authors propose a genetic algorithm for the clustering phase and an insertion mechanism for the routing phase, and the main objective is the minimization of vehicles used. [17] proposed a hierarchical objective function. The algorithm first maximizes the number of served users and then minimizes the inconvenience expressed as the linear combination of waiting time and excess ride time.

The most important thing about the dial-a-ride problem is that the problem is $\mathcal{NP}$-hard. Subsequently, the complexity of the algorithms to find the exact solution is exponential and becomes computationally intractable by increasing the number of parameters, including the number of requests and vehicles. As we observe here from reviewing the literature by 2007, the DARP is only computationally tractable for few hundreds of requests in the static case. In practice, DARP only implemented in the cases with a few thousand instances, namely, in elderly transportation. Developing efficient algorithms that can deal with the dynamic case and provide adequate quality solutions in a short time for the problem with large instances is a must for handling real-time dynamic ridesharing.

To cover the recent development in the dial-a-ride problem since 2007, we start with the objective function. As discussed before, the most popular objective functions are concentrated on minimizing the operational cost (e.g., total traveled distance, total transportation time, number of vehicles required) and users' inconvenience metrics (e.g., detour from the shortest path, users' waiting time, deviation from time windows provided by users) [47]. We observed a shift in the objective function since 2007, which concentrates more on the environmental aspect and the quality of the service for passengers. For example, [5] consider a static version of the DARP, in which all requests are known in advance, and each request consists of the number of persons to transport, the pickup and delivery locations, and time windows. For the objective function, they consider a combination of the transportation cost, the user's dissatisfaction, and the quantity of CO_2 emitted by all the vehicles. [38] added the occupancy rate to the objective function and maximized it. They proposed two approaches for finding the exact solution for up to 200 instances. [66] consider a static decision model for DARP with a fleet of autonomous electric vehicles and try to maximize the system reliability by minimizing the number of loading/unloading operations.

[47] reports that only a few numbers of researches have proposed solution methods to stochastic or dynamic DARPs. [80] study a stochastic and dynamic version of DARP; they consider the changes in travel speed caused by traffic situations and accidents and then conclude that eliminating this information from the model could lead to poor user satisfaction as it missed time windows. They tested their approach on instances with up to 762 requests and observed that exploiting stochastic information about travel speed can significantly improve user satisfaction. [51] considers a dynamic variant of DARP where trip requests are immediate. They modeled a single-vehicle problem using a Markov decision process and developed a non-myopic control policy for the dial-a-ride problem. [97] considered dynamic DARP over complex constraints and proposed a flexible scheduling scheme to cope with fluctuation in trip duration, new requests, vehicle breakdowns, and different stochastic events. Furthermore, they provided a fast heuristic to reoptimize the solution in the occurrence of an event. The dynamic version of DARP is still unable to provide fast and efficient solutions to the routing and scheduling for real-life large-scale requests, which can exceed hundreds of thousands of requests daily.

2.2 Ridesharing Problem

Here we review recent works in the ridesharing problem. [4] defined ridesharing as a single or recurring rideshare trip without any fixed schedule, where the matching process started in a short time before the trip began. The authors aimed to identify and discuss the obstacles and benefits of real-time ridesharing. They recognized the following technological requirements for ridesharing: They consider smartphones with constant network access and having GPS as the key enabler on the real-time ridesharing. By accessing high-speed internet on the phones, frameworks can provide real-time information for passengers and also develop location-aware platforms to use the GPS data in order to arrange the ridesharing. A ridesharing platform needs a ride-matching algorithm to match a rider to a driver to complete their trip together. The matching process needs to find a solution in a short time for a large number of instances. It is important to note that in [4], authors consider ridesharing as the process in which a rider and a driver share their ride and accomplish their trips together. Ridesharing can take a variety of definitions based on the system design, and we are going to study them here.

[4] identified challenges associated with ridesharing and categorized them into the following categories; Economic challenges, social and behavioral challenges, institutional challenges, and technological challenges. we summarize their identified challenges as follows:

1. Economic challenges
    - Imperfect information.
    - High transaction cost.
    - Subsidies favoring other transport modes.
    - Public tendency toward vehicle ownership.
2. Social and behavioral challenges
    - Stranger Danger.
    - Need for mutual dependency.
    - Reliability of service.
    - Flexibility in schedule.
3. Institutional challenges
    - Insufficient institutional collaboration.
    - Business and revenue model.
    - Competition of services within a market.
4. Technological challenges
    - Measurement of successful rideshare trips.
    - Integration of multimodal travel information

Companies like Uber and Lyft, in practice, tackle some of the most critical challenges in ridesharing, specifically in economic and social and behavioral challenges. Automated system reduces the transaction cost, user profile, and rating systems reduce stranger danger and imperfect information. [2] provides the problem characteristics for dynamic ridesharing by introducing the key features of the problem. [2] consider the following six features for dynamic ridesharing: Dynamic ridesharing can be established on short notice, and the range can vary from few minutes to few hours. [2] considers ridesharing as a system in which the drivers are independent and different from most transitional vehicle transportation, in which they had a fleet of vehicles. Cost-sharing is one of the main features in the ridesharing system, and the cost allocation should be beneficial for everyone engaged in ridesharing in terms of reducing travel costs. Dynamic ridesharing focuses on single, non-recurring trips, and it distinguishes dynamic ridesharing from the carpooling model, which requires long term commitment between riders and drivers. In a ridesharing system, the assumption is that participants agree to share a ride in advance. The last feature is the ability of ridesharing for automated matching. Automated matching in ridesharing can reduce the efforts for participants, and based on the definition of [2], the matching process in ridesharing is a bipartite matching between riders and drivers.

The comparison between DARP and ridesharing based on their features clarify the differences; First, ridesharing focuses on an agile system with shorter response time to trip request comparing to DARP, second, in ridesharing drivers are independent. On the other hand, both of the DARP and ridesharing are common in the objective function. Like the DARP problem, most studies on ridesharing consider minimizing system-wide vehicle miles, minimizing system-wide travel time, and maximizing the number of served requests. As mentioned earlier, the ridesharing system aims to respond faster. As an example, in DARP, time constraints are modeled as time windows, and in ridesharing problem drivers and riders indicate the departure time. In ridesharing systems, one of the motivations for both sides of the market (riders and drivers) is to reduce their trip cost, and cost constraints play an important role in modeling.

Constraints control the quality of the matches and play an essential rule in any optimization problem. [2] introduces the time constraints as an essential consideration for matching riders to drivers. Different approaches are proposed to deal with trip schedule preferences; many of the available services ask for the desired departure time. Some others use the same approach as mainly used in the DARP and ask participants to define a time window. [1]

lets participants specify the earliest departure and latest arrival time. The other approach is to indicate the tolerance for a detour from the shortest path; as an example, [4] asks participants to specify the maximum excess for travel time willing to accept. The other important consideration is related to trip expenses. Both sides of the market (riders and drivers) are willing to reduce their trip cost by sharing their ride. There are various ways to divide the cost between riders and drivers. [40] suggest dividing the shared part of the trip between the participants. [1] propose a way to allocate the cost proportional to the distance of the separate trips.

[2] considers the dynamic ridesharing as a system where new riders and drivers enter and leave it. Both of the riders and drivers announce a planned trip as they enter the systems, the difference is drivers *offering* a ride and riders *requesting* a ride. As new drivers and riders continuously arrive, the system has incomplete information to decide on matching the driver to the passenger, and different approaches are taken by researchers to address this issue in dynamic ridesharing. For example, [1] used a rolling horizon to deal with planning uncertainty; in their approach, the optimization problem was solved for all the available rides. They run the algorithm for finding rideshare arrangements each time a new ride comes to the system. [98] consider a multi-agent simulation for ridesharing to establish rideshare locally; one of the prominences of their work is that drivers and riders can be matched en-route. One ability that has recently become available is the anticipation of future requests. In dynamic planning literature, several works show that incorporating future requests and stochastic information can lead to an improvement in decisions [11, 67, 81].

One of the most exciting and recent works in dynamic ridesharing is [90]. They conduct an extensive computational study to identify the impact of different participants' flexibility on the performance of a single rider, single driver ridesharing systems. In their ridesharing platform, they receive a set of rideshare announcement S over time. S consists of two disjoint subsets R and D, which represents the requests for riders and drivers, respectively. For each $s \in S$, they had an origin location o_s and a destination d_s for the trip spatial information and a_s denotes the announcement time and l_s denotes the latest arrival time at their destination and e_s denotes the earliest departure time and carries the temporal information. d_{ij} and t_{ij} denote the distance and travel time between location i and j, respectively. In their setting, a driver $i \in D$, travels alone if not matched to riders by the latest possible departure time $l_i - t_{o_i d_i}$ and they do not allow en-route matching, and subsequently, a match needs to be established before the driver's depart from o_i. Their primary objective is to maximize the number of matched participants. As in [1], They formulate the problem as a weighted bipartite matching problem, and as [91] used a hierarchical optimization approach to first, maximize the number of matched participants and subsequently maximize the system-wide vehicle miles saving.

The different types of flexibility that they consider in their modeling for ridesharing includes matching flexibility, scheduling flexibility, and detour flexibility. [90] defines the matching flexibility as the willingness of participants to depart earlier or later to be able to find a match and for a trip $s \in S$ formalize it as the difference between latest arrival time l_s and the earliest departure time e_s and, the direct travel time from origin to destination, $f_s = (l_s - e_s) - t_{o_s d_s}$. scheduling flexibility refers to available time in order to find a match. For a trip $s \in S$, scheduling flexibility is the difference between the latest departure time $l_s - t_{o_s d_s}$ and the announcement time a_s. Because en-route matching is not possible in their model, participants can be matched between the announcement time and the latest departure time. The scheduling flexibility in their model is composed of two parts: the matching flexibility and the announcement lead time $(e_s - a_s)$. Detour flexibility is the tendency of drivers to make a detour to accommodate riders, and it is proportional to the direct trip duration plus the service time. Here we provide an example for illustration in fig2.1 and provide a trip timeline. Let suppose a participant announcing a trip with a lead time equal to 15 minutes, who wants to be at the destination within one hour from the request time with a direct travel time of 35 minutes; it implies scheduling flexibility of 25 minutes and matching flexibility of 10 minutes.

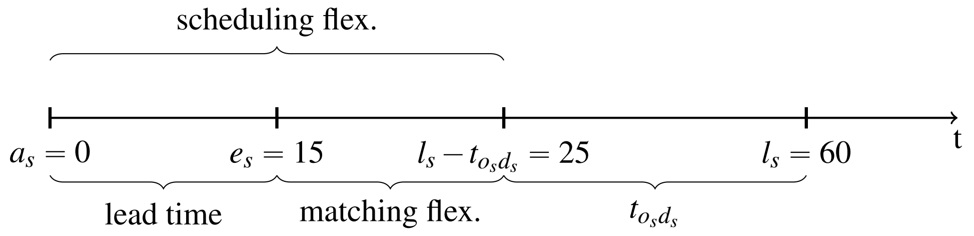


FIGURE 2.1: Scheduling and matching flexibility for a ride request in [90]

Here we describe numerical results by [90]. Their experiments were done in an offline mode; in other words, they have had complete information. They concluded that the matching rate in the offline mode could provide an upper bound on the matching rate when dealing with the online mode. Two sets of instances were considered for the numerical experiments, namely, urban area instances, and corridor instances. The number of requests varied from 500 to 5000. In corridor instances, the trips only had one direction, whereas the trips' directions were random in the urban area. The other difference between these two sets of instances was that the occurrence space was relatively more abundant in corridor instances (20 by 6 miles) comparing to urban area instances, which were generated on a 6 by 6 miles square. To observe the effects of the different flexibility, they ran their model for ten different sets. They, therefore, reported the average performance considering different values for matching flexibility, detour flexibility, scheduling flexibility, and finally, the impact of density on the performance of the model.

They increased the matching flexibility from 5 min to 60 minutes for both the corridor and urban trips. They then observed that low system-wide matching flexibility could limit the ability to find a match even for the highest system density, which is 5000 trips per day. The matching rate for 5 minutes matching flexibility in corridor instances started with 6.9% for 500 requests and increased to 30.2% for 5000 requests. For the urban area instances, with the same matching flexibility, the matching rate started with 12.8% for 500 requests and increased to 46.7% for 5000 requests. They concluded that the reason for this is that low matching flexibility only allows matches between trips with similar spatial and temporal identity. Higher matching flexibility could increase the matching rate. For example, in their experiment with 30 minutes matching flexibility with the lowest density (500 requests), the matching rate increased to 62.8%, and for the highest density (5000 requests) it increased to 90.6% for corridor trips and 55.9% and 84.9% for the urban area, respectively. They settled that a moderate level of matching flexibility would be beneficial to increase the system-wide matching rate, but the system does not gain that much by increasing the matching flexibility beyond a moderate number.

They also observed that the matching flexibility patterns are different for riders and drivers. They explained it by the fact that riders did not have to perform detour, and their matching flexibility only represented their waiting time; however, for drivers, this matching flexibility captured both the waiting time and detour from their shortest path. They noted that even for 1000 trips (which is relatively small for transportation requests), approximately 45% of the riders could pick up within 2 minutes of matching flexibility, and the rest needs could be addressed with a matching flexibility range from 2 to 12 minutes. It is also important to note that the most crucial factor in achieving a high matching rate is the number of trip announcements called ”critical mass” in sharing economy. For example, to achieve a 90% matching rate with 2500 requests, the matching flexibility should be 60 minutes, but for 5000 requests, a 90% matching rate would be achievable by 40 minutes of matching flexibility. To put this paragraph in a nutshell, we conclude that, to find a good match, we need to increase the number of possible matches, for which we have two options. First, increasing the waiting time which subsequently leads to a more extensive pool of requests and also increases the matching possibility, second, raising the number of requests which leads to a dense pool of requests even in a short time so one can find a good match with a relatively short waiting time.

The next parameter that [90] vary to see the impact on their sharing model is detour flexibility. As described before, detour flexibility in their setting only affects the drivers because riders move directly from their origin to their destination. They vary the system-wide detour flexibility between 5% to 50% for fixed matching flexibility of 20 minutes for all of the participants. They observed that for fixed detour flexibility, the matching rate and number of requests were positively correlated. In other words, by increasing the number of requests, the matching rate increased, but the marginal increases declined. Based on their numerical results, it is essential to note that drivers' compliance to increase their detour flexibility leads to an increase in the matching rate substantially. For example, increasing detour flexibility from 20% to 50% for 5000 requests in corridor trips could increase the matching rate from 80.8% to 97.3%. They also note that, even by increasing the overall detour flexibility, the detours in numerical experiments appear reasonable from the drivers' inconvenience perspective. It is one of our motivations for considering an autonomous fleet instead of human drivers, and in the next section, we address the other advantages of automation in distribution and routing over the human.

As illustrated in fig 2.1, scheduling flexibility is defined as the sum of the matching flexibility and announcement lead time. In qualifying the importance of scheduling flexibility, [90] observed that announcement lead time had an important role in the matching rate. They also reported that a combination of short lead time and small matching flexibility could significantly limit the ability of the model to match the riders to drivers, and in the case in which the matching flexibility was high, announcement lead time did not have a significant effect on the matching rate. Based on their results, the authors suggest that dynamic ridesharing may not be feasible in practice with short lead time and low matching flexibility. They increased the number of request from 5000 to 20000 and observed that it is almost impossible to achieve more than 60% of matching rate on zero lead time and with

5 minutes matching flexibility and 5 minutes lead time it becomes possible to match up to 70% of the participants in the system.

In [61], the authors used NYC taxi data to evaluate their ridesharing system. Their main contributions are: (1) proposing a model which considers heterogeneous preferences of the passengers; (2) they compared traditional taxis to autonomous taxis, and (3) they examined the spatial changes in service converge after adopting sharing. Their experimental results indicated that by adopting ridesharing, the occupancy rate increased from 1.2 to 3. The total travel distance decreased by up to 55%, and the carbon emission was reduced by 866 metric tons per day.

2.3 Mobility on Demand Problem

[3] considers mobility-on-demand (MoD) as a new mode of transportation led by companies such as Uber, Lyft, and Via as the consequence of large-scale adaptation of smartphones and the decrease in cellular communication costs. MoD systems can provide a reliable mode of transportation for their users. In their modeling, they consider a fleet of vehicles with varying passenger capacities and address both vehicle assignment to request and repositioning the fleet to service demand. They provide an online algorithm that converges to optimal assignment over time and starts with a greedy solution for the initial start.

For a request r, they defined the total travel delay, which included both of the waiting times and detour from the shortest path, and their objective was to minimize the total travel delay system-wide for served requests plus a large penalty for unserved requests. Their procedure to assigning requests to vehicles consist three steps: In the first step they create the RV-graph in which the nodes are requests and vehicles, two requests r_1 and r_2 are connected if a virtual vehicle could pick up and drop off them while satisfying a set of constraints and a value equal to the sum of their total travel delay assigned to the edge weight. A vehicle v and a request r are connected if the vehicle v could serve the passenger r under a set of constraints. The RTV-graph is created in the second steps, which connects a trip (a set of requests) to vehicles or requests and contains two types of edges: edge $e(r,T)$, which connects a request r to a trip T that contains r, and edge $e(T,v)$ between a trip T and a vehicle v that can execute the trip T. They formulated the problem as an integer linear programming (ILP) problem on the RTV-graph and solved the ILP incrementally from a greedy assignment and improve the quality of the solution over time.

To assess the MoD platform's performance using the proposed algorithm, they used real data from NYC for a week from 0000 hours Sunday, May 5, 2013, to 2359 hours, Sunday, May 11, 2013. They considered the complete road network in Manhattan with the mean travel time as the weight of the edges. Each day of experiments contained between 382,799 for Sunday and 460,700 for Friday requests. The other parameters which varied in their experiments were vehicle fleet size that ranged between 1000 and 3000, vehicle capacities that varied between 1 to 10, a maximum waiting time between 120 to 420 seconds, and the maximum trip delay between 240 to 840 seconds. In other words, they considered vehicle detour equal to waiting time. Their experimental results showed that rebalancing the vehicle during the model's execution increases the matching rate by about 20%. Other phenomena that they observed is that, for a fleet with 1000 vehicles with ten seats, during peak time (1800) of the Friday, 10% of the vehicles had eight or more passengers, 40% of the vehicles had six or more, and 80% of the vehicles happened to have three or more, and only 2% of vehicles had only one passenger. They also observed that by increasing the waiting time and total travel delay as expected, the mean of the passengers on board increases. The in-car travel delay would increase with the rise in vehicle capacity as expected. They reported that increasing the vehicle capacities not only boosts the service rate but also reduces the mean distance traveled by vehicles in the fleet, which is the main factor that contributes to pollution, cost, and congestion. As this study presented, the autonomous vehicle fleet can reduce the pickup time by the efficient distribution of vehicles in the coverage area and also increase the chance of finding a match by reducing the waiting time.

In [62], authors considered the dynamic taxi ridesharing, and they provided a fast approach to search the taxis which could satisfy a ride request, and afterward, their scheduling approach inserted the request to the taxi schedule. They ran the proposed approach on real data from Beijing and observed that their approach increases the number of served passengers up to 25% with the same number of taxis, and they also noted a 13% saving in total distance traveled by vehicles compared to the no-ridesharing setting.

[64] is the most recent survey on models and algorithms on the general problem of shared mobility. They considered a vast set of problems, including people sharing rides and people and goods sharing rides. They reviewed the recent researches since 2010 on

ridesharing, carpooling, vanpooling, DARP, and shared-taxi problems. Here we briefly explain their review on people sharing rides. They identified that an insignificant number of researches considered cost restriction during the matching procedure and recognized it as an exciting avenue for research to focus more on individual traveler instead of concentrating on the system-wide objective. Another exciting part of [64] is the ridesharing with autonomous vehicles. They introduced autonomous vehicles (AVs) as an emerging technology that is expected to bring fundamental shifts in vehicle modes of transportation. [88] described humans as drunk robots when the autonomous vehicles effectively work and regulate and indicate the emergence of the shift towards driverless vehicles, especially while large mobility providers (Tesla, Ford, Lyft, Uber) have announced their plan to deploy autonomous mobility services. [57] showed that by deploying autonomous vehicles, the fleet size would significantly decrease compared to the conventional vehicles to serve all the travelers. Deploying autonomous vehicles could reduce traffic congestion, increase travel speed, and reduce travel time besides environmental benefits, which is subsequently achieved by reducing the number of vehicles in the street.

2.4 Advances in Related Fields

Here we briefly review the advances in machine learning and its application on ridesharing and, more broadly, the transportation system. We begin by explaining recent advances, especially over the past ten years, in machine learning for the following topics.

- To solve the combinatorial optimization problems using machine learning.
- To predict future transportation demand.
- Reinforcement learning advances in transportation.

Before discussing the solving of combinatorial optimization problems using machine learning, we discuss an interesting problem in online decision making. In 2017, [26] defined a variant of the celebrated k-server problem, namely, the stochastic k-server problem or how should Uber work, which is an interesting problem and we will explain it here. In the general form of the k-server problem, we have k servers on a metric space $\mathcal{M}$, followed by receiving an online sequence of n requests where the i^{th} request is a point r_i in the metric space $\mathcal{M}$ ($r_i \in \mathcal{M}$). After receiving a request r_i, a server needs to move from its current location to r_i at a cost equal to the distance traveled, and the system-wide objective is to minimize the total cost of serving all the requests. In the stochastic variant of the k-server problem they suppose n independent distributions $< P_1, P_2, \ldots, P_n >$ are known in advance and a request r_i is drawn from P_i. They also define *Uber*-problem where each request r_i consists of two points, namely source and destination in metric ($r_i = (s_i, d_i)$ where $s_i, d_i \in \mathcal{M}$). Serving a request is to move a server from its current location to the source and afterward move towards the destination, and the objective is to minimize the total distance traveled by the server. They obtain a 5-approximation algorithm for the Uber-problem in line and circle metric and $\mathcal{O}(log\ n)$-approximation algorithm for general metrics.

The operation research field came to being during the second world war to assist the military planners in their decision using mathematics and computer science [33]. Nowadays, operation research is expanded to a wide range of humankind's planning, including transportation, supply chain, finance, and economics [10]. The computational complexity of the problem in discrete optimization varies from polynomial time, for example, in shortest path problem to $\mathcal{NP}$ problems, which is unlikely that an algorithm with polynomial running time exists concerning the size of the problem. An example of this combinatorial problem is $\mathcal{NP}$ called the traveling salesman problem (TSP) and is therefore defined as follows: Let suppose we have a collection of n cities, namely $C = \{c_1, \ldots, c_n\}$ and the travel cost between each pair of them. The TSP problem aims to find the cheapest way to visit all the cities and return to the starting location. In 2016, [9] presented a framework to solve a combinatorial optimization problem with a focus on TSP using neural network and reinforcement learning. Their approach could find near-optimal results for 2D Euclidean graphs with up to 100 nodes. They also applied their approach to the knapsack problem and obtained optimal solution for instances up to 200 items. Designing heuristics or approximation algorithms for $\mathcal{NP}$ combinatorial optimization problems requires significant specialized knowledge and [25] proposed a learning approach using a unique combination of reinforcement learning and graph embedding to exploit the structure of the optimization problem. They used the proposed framework on a diverse range of combinatorial optimization problems, including Minimum Vertex Cover, Maximum Cut, and TSP, and showed that their framework learns effective algorithms for mentioned problems. It is important to note that when deploying a learning-based method, these learning procedures would learn based on the sample they observe during the training phase, and to achieve good results (close to optimal solution) during the test phase the problem structure cannot change [10]. In other words, the current state-of-the-art work in machine learning has a limited ability for generalization in combinatorial problems. Instance size and structure cannot differ due to the limited generalization in the learning model. For example, if we train the model to solve the TSP problem in Montreal, there is no guarantee that the model works well for San Francisco, and it is a must to use the trained model locally.

[11, 67, 81] reported that in the dynamic case of ridesharing, considering the stochastic information and predicting upcoming requests led to an improvement in planning and attaining better results. Here we review the state-of-the-art learning method for predicting future demand. As a result of a considerable amount of available data from different platforms, the current models mainly use a mixture of different neural network classes, especially recurrent neural network (RNN) to capture the temporal relation and convolutional neural network (CNN) to capture the spatial relation. [99] divided the New York City area into a grid and trained an LSTM model for each grid to learn the demand pattern and indicated that their approach outperformed the naive statistics average and feed-forward neural network according to RMSE as the evaluation metric. [95] proposed DeepSD, an end-to-end framework to predict the gap between supply and demand to rebalance the vehicles and improve the overall user experience. They discretized the time into 1440 interval (1 minutes interval) and divided the city map into a grid network and defined the *supply-demand gap* at time t in a tile as the total amount of missed requests. They used the embedding technique to concatenate the time, area, and week information as the identity block and weather and traffic as the environment block and subsequently used a fully connected layer with 32 neurons and a single neuron with Relu activation function as the output. They evaluated the performance of the model on DiDi dataset and outperformed LASSO, Empirical Average, and Gradient Boosting Decision Tree based on MAE and RMSE as the evaluation metric. [100] used a mixture of CNN, RNN, and structural embedding to predict taxi demand.

The authors used a multi-layer CNN to capture the spatial dependency among neighbor regions, and to capture the temporal dependency, they used an LSTM model, which took the representation of CNN and concatenated with external features like traffic data and weather. In the embedding part, they created a weighted graph of regions where the weighted edges represented the functional similarity, and the concatenation of embedding and LSTM representation was used for demand prediction. The external information can play a significant rule in predicting the demand and capture the underlying trend in the system dynamic. As an example, during the Covid-19 pandemic, the number of Uber requests and, more generally, the transportation requests were significantly reduced due to changes in human behavior. This phenomenon occurs in daily life; for instance, events can change the location demand, and considering these kinds of information could help the model to reduce the error. In [77], authors proposed to combine the time-series demand data with textual data to predict the demand in event areas. In their proposed framework, textual data is embedded, and feature extraction is done by a 1D convolutional layer, and the representation is combined with the time-series output as the input of a fully connected perceptron. They observed that combining these two forms of the data can significantly reduce the demand prediction error in events that reveals an irregular pattern.

In [99, 95, 100, 77], demand is reduced to the origin of the trip but, we can define a more accurate definition for demand in transportation by defining demand with the origin and destination. [60] proposed a deep learning framework for taxi origin-destination demand prediction. In their work, two convolutional neural networks trained to capture spatial dependency for both origin and destination. Furthermore, they trained an LSTM network to capture the temporal dependency. A global correlation context is used to unite the origin and destination prediction. As we described above, considering future demand had a significant effect on the planning and scheduling of transportation problems. The massive amount of data and powerful models with the ability to capture the dynamic of the environment can reduce the gap between offline and online models in ridesharing. During the past ten years, by gathering massive data by online ride-hailing platforms like Didi, Uber, and Lyft and advances in learning algorithms and optimization methods, training accurate and robust models to predict the future has become possible and can be used in planning.

Reinforcement learning (RL) is a sub-field in machine learning. In RL, an agent interacts with an environment through a Markov decision process (MDP), and the interaction can be summarized as follows: At every time step, the agent is in its current state and chooses an action according to its policy. The agent, based on the state and action, transits to a new state and receives a reward, which is the environment feedback to the agent. The goal in RL is to train the agent to maximize the expected sum of future reward [10]. Here we review the recent works in transportation which have used RL to learn better policies for the vehicles routing during the idle (cruising) time or serving a passenger.

In [48], the authors proposed a deep reinforcement learning approach for vehicle dispatching and repositioning problem. In their work, the agent represents central fleet management, and it controls all the vehicles. They studied both the driver-centric and system-centric reward formulation in their empirical studies and presented that the system-centric approach is superior in maximizing long term rewards. Authors in [58] used a cooperative multi-agent reinforcement learning for resource balancing in a complex logistics network. They modeled the resource balancing problem as a stochastic game $\mathcal{G} = (N, \mathcal{A}, S, \mathcal{R}, \mathcal{P}, \mathcal{Y})$, where N is the agent set, $\mathcal{A}$ is the joint action space, S is the state set, $\mathcal{R}$ is the reward function, $\mathcal{P}$ is the transition probability function, and $\mathcal{Y}$ is the discount factor. They ran extensive experiments for evaluating the performance of the proposed model and observed that their approach could stimulate the cooperation among agents, which were vehicles reported to have significant improvement in both performance and stability due to cooperative setting. Automate decision making, in this case, using RL to improve drivers' behavior, leads to improving the overall performance of the system. Reinforcement learning is also used in the transportation system to control more than the cruising time of vehicles. For example, in [56], the authors used deep reinforcement learning to maximize the revenue by assigning the vehicles to requests, recharge them, and lastly, reposition the vehicles based on future anticipation. In [45], authors focused on the coordination in MoD services and formalized the adaptive coordination of vehicles using a reinforcement learning framework. By experimental analysis on large-scale datasets (21 million rides from Uber, NYC yellow-cab data, and Didi), they observed that their approach reduced the request rejection rate and waiting time significantly. [86] argued that the underutilization of taxi resources due to taxi drivers' behavior during their idle time would lead to a high rejection rate since the drivers were not in the correct place to take the upcoming passengers. They proposed a bilevel optimization with the upper level as reward design and lower level as multi-agent reinforcement learning where the upper level interacts with lower level by adjusting rewards. Based on their experiment on NYC yellow-cab data, the authors reported that their approach increased the revenue by up to 4%.

[63] considered the taxi dispatch problem for autonomous vehicles using a deep reinforcement learning approach. They proposed a model-free actor-critic algorithm with a feed-forward neural network to approximate both policy and value function. Their numerical studies showed that their approach converged to a theoretical upper bound with less than a 4% optimality gap. [59] proposed a contextual multi-agent reinforcement learning

framework including a contextual deep Q-learning and contextual multi-agent actor-critic to tackle large-scale fleet management problem. In their problem, they considered a large set of homogeneous vehicles are available for the ridesharing platform, and their goal was to maximize the value of all the orders served by repositioning the vehicles to the areas with a larger demand-supply gap. [94] focused on taxi navigation during their cruise time (when there is no customer onboard) to maximize taxi drivers' revenue, and therefore proposed a reinforcement learning framework. Their experiment on real-life data showed that agents were capable of achieving revenue comparing to the top 10 percentile of drivers.

Chapter 3

Online Ridesharing

As discussed in the introduction, ridesharing can take various forms on how to serve the passengers. One of the most important aspects of ridesharing is the knowledge of the system about the upcoming request. Here we consider an online version of ridesharing which decision-maker does not have any prior knowledge about the requests and requests come over time from an unknown distribution. In this work, we propose a ridesharing platform for urban transportation. The characteristics of urban transportation must be considered in order to satisfy the passengers. In urban transportation, trips are usually short in distance and not planned. We consider the characteristics of urban transportation in our ridesharing design. In a ridesharing system, drivers play an essential role, and their behavior could change the system performance significantly. [3] shows the improvement of the quality of the rides using autonomous vehicles. In this work, we also consider the decision-maker has a fleet of autonomous vehicles and has the optimal policy for controlling them to serve the requests. As mentioned earlier in the introduction, we only consider the passengers in this work, and finding the optimal policy for vehicle routing and repositioning is out of context.

We aim to minimize traveling costs for passengers within a few minutes from their request time by matching passengers to each other to accomplish a ride together and share the trip expenses. Our proposed method works for one day from 00:00 to 23:59, and during this time horizon, users send their requests for a ride and the decision-maker match up the passenger to share their ride and an autonomous vehicle dispatch to serve their requests. Vehicle transportation provides convenient door-to-door transportation, but one of the drawbacks is the cost compare to alternative transportation modes. Sharing rides has the potential to reduce the trip cost by dividing the trip costs between the participants. In this work, two passengers match each other if the trip cost for both of them reduced. The remaining part of this chapter is as follows: In section 3.1, we introduce the problem in mathematical form. In section 3.2, the waiting time function, which is the key component of our approach for reducing the travel cost for passengers introduced. In section 3.3, we describe our procedure to create a graph for the passengers and the graph used in 3.4, which we introduce two approaches for matching the passengers to each other.

3.1 Problem Definition

We consider a single decision-maker with an autonomous vehicle fleet with enough size to serve the rides. During the time horizon, a set of passengers send their request for a ride in an online manner. Each request r_i consist a tuple (p_i, d_i, t_i) where p_i is the pickup location, d_i is the dropoff location and finally the t_i is the time that request sent to the system. The decision-maker keeps track of the passenger information in real-time, and the decision-maker task is to serve the passenger request in short notice and match the passengers to share their ride if a feasible matching is available.

We define a system-wide fixed flexibility factor, ε, which models the overhead in travel time. For example, if the $\varepsilon = 0.6$ and the trip duration on the shortest path for passenger r is 10 minutes, the trip duration should be less than 16 minutes if he/she share a ride with another passenger. The motivation behind the ε is to extend the trip duration for ridesharing concerning the trip quality. It is also important to note, by using ε, the amount of temporal information that a passenger needs to provide minimizes. In this work, we consider all the passengers tend to share their ride with other passengers. In our setting, passengers share their ride with another passenger and travel alone in the worst case. Note that passengers travel alone if there is no match for them to share their ride when the matching algorithm run.

Moreover, once a trip started, matching is no longer allowed. So, the passenger knows about their trip cost based on their matched passenger before they get into a vehicle and start their ride. We consider an efficient fleet management system and enough fleet size to provide the vehicle at request location based on our needs. We suppose that the fleet management system can provide a vehicle to a passenger when the waiting time elapsed. In other words, the passengers do not wait for a vehicle to arrive. In this work, we put the fleet management system aside and design a system to handle the ride requests in an online manner.

In this model, the decision-making system had two components:

1. How long to make each passenger wait to find a match for sharing the ride when the waiting time elapsed.

2. How to match the passengers based on their pickup and dropoff location.

The first component will be addressed in section 3.2, and the second component will be discussed in the section 3.3 and section 3.4. Once passengers are matched, we assume that a nearby vehicle dispatched to fulfill the trip. Based on the future demand, the waiting time function finds the time that minimizes the expected travel cost. The passenger graph and matching procedure are used to find the proper match between passengers and thoroughly discuss them in their sections.

The objective of this work is to maximize the percentage of the matched passenger with respect to flexibility constraints, which guaranteed the quality of the trip for the passenger. We define a finite time horizon, for example, one day for each round of our algorithm. The horizon fixed to work from 00:00 to 23:59 as a day. During the horizon, the decision-maker receives a set of request $\{r_1, r_2, ...\}$ with unknown distribution both in time and locations. The decision maker's goal is to maximize the objective of the problem. For simplification of the problem, we replace the actual city network with a grid graph. Suppose we have an $m \times n$ grid graph, which has $(m+1) \times (n+1)$ vertices and $2nm+n+m$ edges, which represent the road and weights represent the shortest travel time between the point at time t. The edges weight are **dynamic** and represents the travel time at time t and represented by $G_{m,n,t}$.

By replacing the actual city network by a grid graph, we first explain the representation of a request in the grid graph and afterward we provide some definitions for the transportation in grid graph. At time t, each request $r_i = (p_i, d_i, t_i)$ that come to system at time t had the following property:

- $t_i = t$

- $p_i, d_i \in V(G_{m,n,t})$

Two vertices v and w in $G_{m,n,t}$ are neighbor if $e(v,w) \in E(G_{m,n,t})$. When passenger r_i, request for a ride from p_i to d_i, we define a valid travel path a follows:

Definition 3.1.1. (Valid Trip Path)
For a request $r_i = (p_i, d_i, t_i)$, a valid trip path ϕ is a sequence of vertices $v_1, ..., v_k$ where $v_j \in V(G_{m,n,t})\ \forall j \in \{1, \dots, k\}$ and $e(v_j, v_{j+1}) \in E(G_{m,n,t})\ \forall j \in \{1, \dots, k-1\}$ and $v_1 = p_i$ and $v_k = d_i$.

For a valid trip path ϕ, the trip duration for the path denoted by $|\phi|$ and calculated as the sum of weight of edges in $G_{m,n,t}$. We define the shortest trip path for the request r_i as valid trip path which has the minimum travel time and denoted by ϕ_i in the remaining of this work. More generally, for traveling from location $p \in G_{m,n,t}$ to location $d \in G_{m,n,t}$ at time t, $\phi_{p,d,t}$ denotes the shortest path for that trip. To compute the shortest trip path we use Dijkstra's shortest path algorithm on grid graph $G_{m,n,t}$. In chapter 4, we explain our training procedure for regression model using historical data, and in the simulation, the weights are updated using regression model. It is important to note that for a path ϕ, $l(\phi)$ denotes the length of the path and calculated as the sum of the length of the edges (sphere distance between geolocation of the vertices). We assume that each passenger r_i is willing to incur at most ε fraction above $|\phi_i|$ in his travel time to participate in ridesharing. Subsequently, the set of allowable paths for r_i is denoted by:

$$\Phi_i \triangleq \{\phi : |\phi| \leqslant (1+\varepsilon)|\phi_i|\} \tag{1}$$

The set Φ_i contains every path from p_i to d_i, which satisfy the flexibility of the trip. One of the novelties of our approach is to assign an adaptive waiting time for each passenger. The idea behind the waiting time (instead of solving the matching problem online) is as follows: we ask the passenger to wait for a few minutes to increase their chance to share a ride, and when the waiting time elapsed, the matching procedure solves an offline matching problem. To calculate the waiting time, we used future demand to minimize the expected travel cost for the passenger and discuss the objective function for waiting time in the next section. In the remaining part of the section, we describe the preliminaries for the waiting time function.

Let w_i denotes the waiting time for passenger r_i before starting the trip. We assume that autonomous vehicles always show up to start the passenger trip by the end of w_i, and no passenger waits longer because a vehicle arrives late. The waiting time contributes to the overall trip duration, and the following inequality must hold for each passenger r_i entering the system:

$$w_i + \tau_i \leqslant (1+\varepsilon)|\phi_i| \tag{2}$$

Where τ_i is the actual travel time. An important property in the inequality 2 should be addressed, and we explained it with an example. Let suppose for an imaginary passenger r_I, the travel duration is 15 minutes over the shortest path, so in this setting, the $|\phi_I|$ equal to 15 and subsequently the maximum duration of the trip (right-hand side in the above inequality) is equal to 24 for $\varepsilon = 0.6$. Now let us look into the left side of the inequality, the sum of the waiting time and actual trip path after waiting must be less than 24 minutes and an increase in the waiting time w_I cause a decrease in the τ_I, that is it, by increasing the waiting time the number of available passengers in the system increase as the request between t_I and $t_I + w_I$ becomes available. However, as the waiting time for a given passenger increases, the tolerable detour distance decreases, and subsequently, matching can occur between closer itineraries. The optimal waiting time minimizes the travel cost in this regard.

Here we provide some definitions that we used in our modeling. We start by defining passengers pool at any time instance t as Π_t, including all passengers who are still in the system. This set excludes served passengers who have left the system before time t.

$$\Pi_t = \{r_i | t_i \leqslant t \leqslant t_i + w_i\} \tag{3}$$

The passengers pool at time t consists of all the requests that come before t and still waiting in the system. For the passenger r_i, between the request time t_i and the exclude time $t_i + w_i$ the passenger comes in and goes out from the passengers pool and some of the passengers in the pool are proper to share a ride with passenger r_i based on their itinerary and request time. Here we define a primitive definition which only consider the feasibility of the sharing based on the itineraries and do not consider time constraints. primitive definition introduced in our published work [30] and define a compatible match between two passengers.

Definition 3.1.2. (Compatible match)
A pair of passengers (r_i,r_j) forms a compatible match if there exist two path $\phi_{r_i} \in \Phi_i$ and $\phi_{r_j} \in \Phi_j$ which:

$$\exists (a,b) \in \{i,j\} \times \{i,j\} : p_a, d_b \in \phi_{r_i} \cap \phi_{r_j}$$

Two passengers r_i and r_j forms a compatible match, if there exist two path ϕ_{r_i} and ϕ_{r_j} to serve each of them and the intersection of two path includes at least a pickup and a dropoff. The compatible match definition first used in [30] and here we explain it with an example. In figure 3.1, light colors are the pickup locations for two passenger r_{red} and r_{green} and the dark colors are dropoff locations. In other words, passenger r_{red} trip is from node 0 to node

3 and passenger r_{green} trip is from node 1 to node 6. Let assume the edges weight are equal to one in the grid network, and ε is equal to one. In this example, the shortest path for r_{red} marked by red edges in the graph, and the minimum trip duration is 3 minutes. For r_{green}, the shortest path marked by green edges, and the minimum trip duration is 2 minutes.

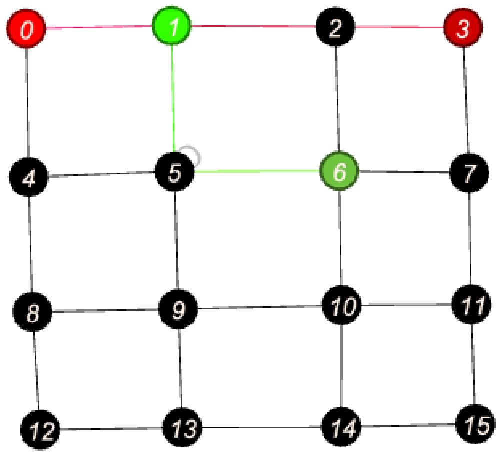


FIGURE 3.1: Disjoint trip path for r_{red} and r_{green}

In this example, based on inequality 2 the maximum overall travel time for r_{green} and r_{red} are 4 and 6 minutes, respectively. Here we show how two passengers r_{red} and r_{green} form a compatible match. In figure 3.2, for r_{red}, the sequence $\phi_r : 0 \rightarrow 1 \rightarrow 5 \rightarrow 6 \rightarrow 2 \rightarrow 3$ is a allowable trip path because trip duration is 5 and less than 6. For the passenger r_{green} the sequence $\phi_g : 1 \rightarrow 5 \rightarrow 6$ is an allowable trip path.

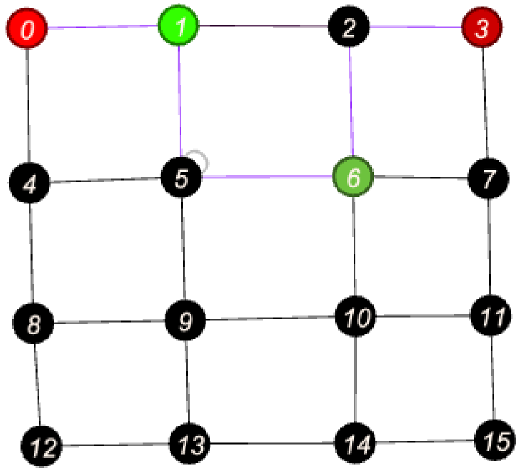


FIGURE 3.2: Travel path after matching occur

Note that for two paths, ϕ_r and ϕ_g, $\{1,5,6\} \in \phi_g \cap \phi_r$ and p_{green} and d_{green} are in $\phi_g \cap \phi_r$. So, the two passenger r_{red} and r_{green} forms a compatible match. It is important to note that we only consider spatial information in the definition of a compatible match and do not consider time constraints in the definition. Here we provide an example to illustrate the drawbacks of compatible match in considering time constraints. In this example, we still have two passengers, r_{red} and r_{green}, r_{red} request for a trip from node 0 to node 15 and r_{green} wants to travel from node 13 to 14. In figure 3.3, the trip path is visualized in the case that passengers travel alone toward their destination.

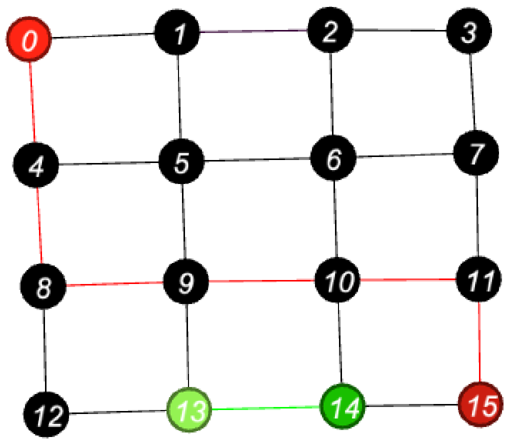


FIGURE 3.3: Deficiency of compatible match

In this example, the travel time on the shortest path for r_{red} and r_{green} is 6 and 1 minutes, respectively. The joint trip path that satisfies the compatible match criteria is visualized in figure 3.4. Here we calculate the trip duration for each passenger, r_{red} moves toward his destination on shortest path and travel time is 6 minutes, but for the other passenger (r_{green}), after the vehicle picked up r_{red}, $r_g reen$ must wait 4 minutes for the arrival of the vehicle and moves toward his destination and the overall trip duration for r_{green} is 5 minutes which is higher than the acceptable detour.

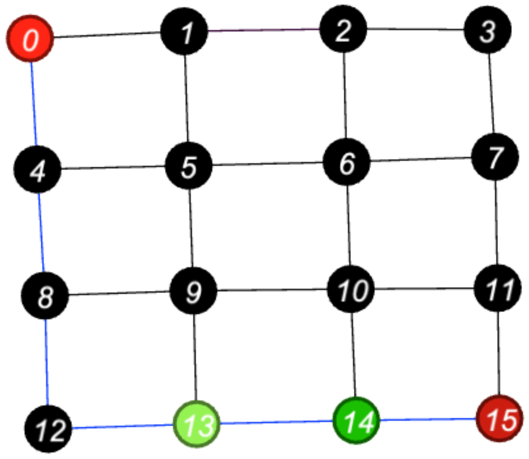


FIGURE 3.4: Overall trip path for r_{red} and r_{green}

Formulating the shared trip path needs to be carefully designed to consider time constraints, and in remaining of this section, we provide the formulation. It is necessary to formulate different parts of a shared trip to measure the passengers' convenience. Here we define the order of sharing to capture the precedence in pickup and dropoff location and calculate the overall trip route when two people share their ride.

Definition 3.1.3. (Order of sharing) For two passengers r_i and r_j, the order of sharing is the set of all possible combinations of pickups and dropoffs to accomplish their trip together.

$$O(r_i, r_j) = \{(p,d) : p \in \{p_i, p_j\}, d \in \{d_i, d_j\}\}$$

Each (p,d) in $O(r_i, r_j)$, indicates the first pickup and dropoff location when r_i and r_j share their ride on that order. In other words, each $(p,d) \in O(r_i, r_j)$ includes information about the overall trip path where a vehicle is needed to serve two requests together. By using $(p,d) \in O(r_i, r_j)$, we are able to calculate the overall travel time for each passenger. For two passengers r_i and r_j at time t, we define the actual trip time function as follows:

$$\tau(r_i, r_j, t, (p,d)) = \begin{cases} [|\phi_{p_i,p_j,t}| + |\phi_{p_j,d_i,t}|, |\phi_{p_j,d_i,t}| + |\phi_{d_i,d_j,t}|] & if\ (p,d) = (p_i, d_i) \\ [|\phi_{p_i,p_j,t}| + |\phi_{p_j,d_j,t}| + |\phi_{d_j,d_i,t}|, |\phi_{p_j,d_j,t}|] & if\ (p,d) = (p_i, d_j) \\ [|\phi_{p_i,d_i,t}|, |\phi_{p_j,p_i,t}| + |\phi_{p_i,d_i,t}| + |\phi_{d_i,d_j,t}|] & if\ (p,d) = (p_j, d_i) \\ [|\phi_{p_i,d_j,t}| + |\phi_{d_j,d_i,t}|, |\phi_{p_j,p_i,t}| + |\phi_{p_i,d_j,t}|] & if\ (p,d) = (p_j, d_j) \end{cases}$$

This function calculate the actual travel time (the time that passenger spend in the vehicle) for passenger r_i and r_j at time t based on order (p,d). It is important to note that,

$\tau(r_i, r_j, t, (p,d))$ is a vector and the first index $(\tau(r_i, r_j, t, (p,d))_1)$ denotes the actual trip time for r_i and the second index $(\tau(r_i, r_j, t, (p,d))_2)$ denotes the actual trip time for r_j. The overall trip path is the path which a vehicle move to satisfied a shared trip between two passengers is denoted by $\Im(r_i, r_j, (p,d), t)$ and is defined as follows:

$$\Im(r_i, r_j, (p,d), t) = \begin{cases} \phi_{p_i,p_j,t} \rightarrow \phi_{p_j,d_i,t} \rightarrow \phi_{d_i,d_j} & if\ (p,d) = (p_i,d_i) \\ \phi_{p_i,p_j,t} \rightarrow \phi_{p_j,d_j,t} \rightarrow \phi_{d_j,d_i,t} & if\ (p,d) = (p_i,d_j) \\ \phi_{p_j,p_i,t} \rightarrow \phi_{p_i,d_i,t} \rightarrow \phi_{d_i,d_j,t} & if\ (p,d) = (p_j,d_i) \\ \phi_{p_j,p_i,t} \rightarrow \phi_{p_i,d_j,t} \rightarrow \phi_{d_j,d_i,t} & if\ (p,d) = (p_j,d_j) \end{cases}$$

The actual trip path and overall trip path for passengers r_i and r_j based on the elements in $O(r_i, r_j)$ explained in the following list.

1. $(p_i, d_i) \in O(r_i, r_j)$:
   - In this case, the trip is started by r_i, and the first person that is dropped off is r_i.
   - The overall trip path is $p_i \rightarrow \cdots \rightarrow p_j \rightarrow \cdots \rightarrow d_i \rightarrow \cdots \rightarrow d_j$.
   - The actual trip path for r_i is $p_i \rightarrow \cdots \rightarrow p_j \rightarrow \cdots \rightarrow d_i$.
   - The actual trip path for r_j is $p_j \rightarrow \cdots \rightarrow d_i \rightarrow \cdots \rightarrow d_j$.
2. $(p_i, d_j) \in O(r_i, r_j)$:
   - In this case, the trip is started by r_i, and the first person that is dropped off is r_j.
   - The overall trip path is $p_i \rightarrow \cdots \rightarrow p_j \rightarrow \cdots \rightarrow d_j \rightarrow \cdots \rightarrow d_i$.
   - The actual trip path for r_i is $p_i \rightarrow \cdots \rightarrow p_j \rightarrow \cdots \rightarrow d_j \rightarrow \cdots \rightarrow d_i$.
   - The actual trip path for r_j is $p_j \rightarrow \cdots \rightarrow d_j$.
3. $(p_j, d_i) \in O(r_i, r_j)$:
   - In this case, the trip is started by r_j, and the first person that is dropped off is r_i.
   - The overall trip path is $p_j \rightarrow \cdots \rightarrow p_i \rightarrow \cdots \rightarrow d_i \rightarrow \cdots \rightarrow d_j$.
   - The actual trip path for r_i is $p_i \rightarrow \cdots \rightarrow d_i$.
   - The actual trip path for r_j is $p_j \rightarrow \cdots \rightarrow p_i \rightarrow \cdots \rightarrow d_i \rightarrow \cdots \rightarrow d_j$.
4. $(p_j, d_j) \in O(r_i, r_j)$:

- In this case, the trip is started by r_j, and the first person that is dropped off is r_j.
- The overall trip path is $p_j \rightarrow \cdots \rightarrow p_i \rightarrow \cdots \rightarrow d_j \rightarrow \cdots \rightarrow d_i$.
- The actual trip path for r_i is $p_i \rightarrow \cdots \rightarrow d_j \rightarrow \cdots \rightarrow d_i$.
- The actual trip path for r_j is $p_j \rightarrow \cdots \rightarrow p_i \rightarrow \cdots \rightarrow d_j$.

The actual trip duration is calculated as the time that a passenger spends in the vehicle when they share their trips based on the order of sharing. When two passengers share their rides to accomplish a trip together, a vehicle dispatch to pick them up based on their sharing order. There is a delay between picking up the first passenger and the second one that should be considered in overall travel time. We defined the vehicle arrival delay for passenger r_i and r_j at time t for order $(p,d) = O(r_i,r_j)$ as follows:

$$\zeta(r_i,r_j,t,(p,d)) = \begin{cases} [0, |\phi_{p_i,p_j,t}|] & if\ (p,d) = (p_i,d_i) \vee (p_i,d_j) \\ [|\phi_{p_j,p_i,t}|, 0] & if\ (p,d) = (p_j,d_i) \vee (p_j,d_j) \end{cases}$$

It is important to note that compatible match definition ignores the time constraints and only considers the spatial constraints in sharing. The time constraints are one of the crucial constraints that should be addressed in ridesharing to meet passengers' convenience. By using the order of sharing, we define a feasible match between r_i and r_j that consider the time constraints.

Definition 3.1.4. (Feasible match) A match between two passenger r_i and r_j at time t is feasible if there exist at least one pair $(p,d) \in O(r_i,r_j)$ in which the following inequalities are satisfied.

$$\begin{aligned} t - t_i + \tau(r_i,r_j,t,(p,d))_1 + \zeta(r_i,r_j,t,(p,d))_1 \leqslant (1+\varepsilon)|\phi_i| \\ t - t_j + \tau(r_i,r_j,t,(p,d))_2 + \zeta(r_i,r_j,t,(p,d))_2 \leqslant (1+\varepsilon)|\phi_j| \end{aligned} \tag{4}$$

It is important to note that t in the definition 3.1.4 is greater than t_i and t_j. The left hand side of both inequalities in 4 calculates the overall trip path including vehicle arrival delay ($\zeta(r_i,r_j,t,(p,d))$), waiting time ($[t-t_i, t-t_j]$) and, actual travel time ($\tau(r_i,r_j,t,(p,d))$) for passenger r_i and r_j. The order of sharing is explained in detail based on the example that we used to explain compatible match and we also discuss the feasibility of the matching for the mentioned passengers. The order of sharing for the mentioned passengers, r_{red} and r_{green}, is the set $O(r_{red}, r_{green}) = \{(p_{red}, d_{red}), (p_{red}, d_{green}),$

$(p_{green}, d_{red}), (p_{green}, d_{green})\}$. We visualize the trip path for each of the mentioned order to see the difference in trip duration.

First, we start with $(p_{red}, d_{red}) \in O(r_{red}, r_{green})$ where the trip path is the sequence $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 6$ which is visualized in figure 3.5.

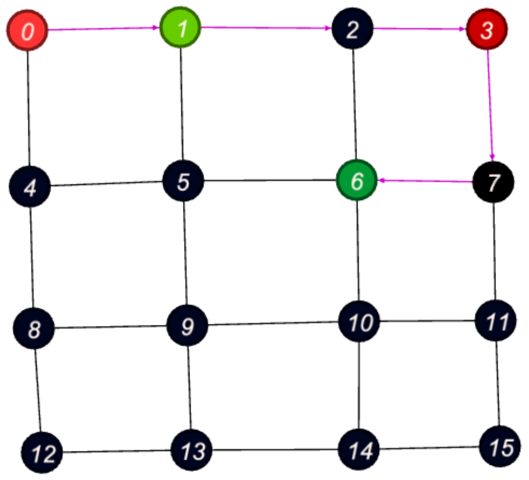


FIGURE 3.5: Overall travel path for $(p_{red}, d_{red}) \in O(r_{red}, r_{green})$

In this case, the actual travel time for r_{red} is 3 minutes, which is equal to the shortest trip from node 0 to node 3, and the vehicle arrival delay is equal to 0. On the other hand, the actual travel time for passenger r_{green} is equal to 4 minutes and the overall travel time for passenger r_{green} from sending a request to dropoff is equal to 5 minutes (including vehicle arrival delay) and does not satisfy 4 inequality.

For $(p_{red}, d_{green}) \in O(r_{red}, r_{green})$, the trip path is the sequence $0 \rightarrow 1 \rightarrow 5 \rightarrow 6 \rightarrow 2 \rightarrow 3$. The overall path is shown with direct edges in figure 3.6.

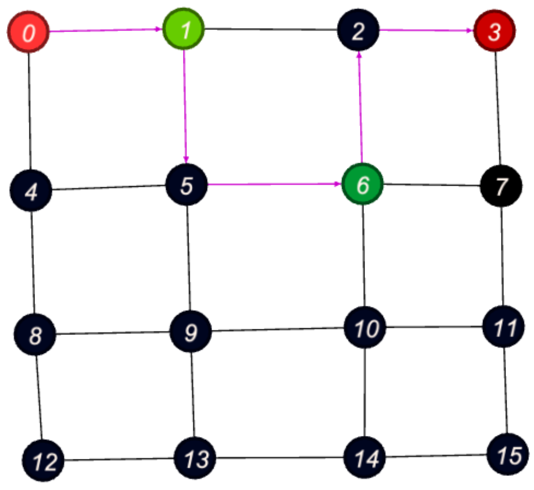


FIGURE 3.6: Overall travel path for $(p_{red}, d_{green}) \in O(r_{red}, r_{green})$

For the overall path in figure 3.6, the overall travel time for the passenger r_{red} is equal to 5 which satisfy the inequality 4. For the passenger r_{green} the actual travel time is equal to 2 and the vehicle arrival delay is equal to 1 (overall trip duration is equal to 3 minutes) which also satisfy the inequality 4. r_{red} and r_{green} form a feasible match on the mentioned order and are able to accomplish a ride together with respect to time constraints.

On the third possible order of sharing which is, $(p_{green}, d_{red}) \in O(r_{red}, r_{green})$, the overall travel path is $1 \to 0 \to 1 \to 2 \to 3 \to 7 \to 6$. The overall trip path is shown in figure 3.7.

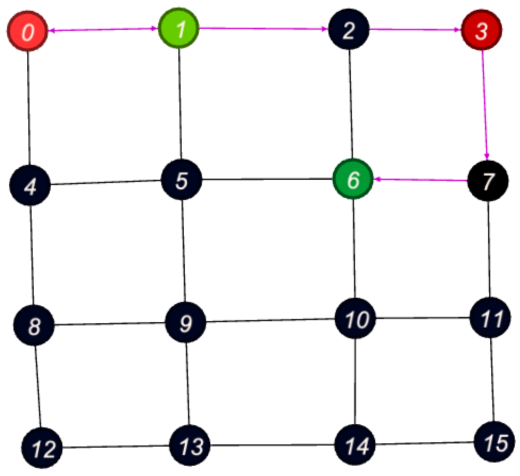


FIGURE 3.7: Overall travel path for $(p_{green}, d_{red}) \in O(r_{red}, r_{green})$

The overall travel time for passenger r_{red} is equal to 4 and for the passenger r_{green} is equal to 6 which is not satisfy the inequalities 4.

For $(p_{green}, d_{green}) \in O(r_{red}, r_{green})$, the overall travel path is $1 \rightarrow 0 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 2 \rightarrow 3$. The overall trip path is shown in figure 3.8.

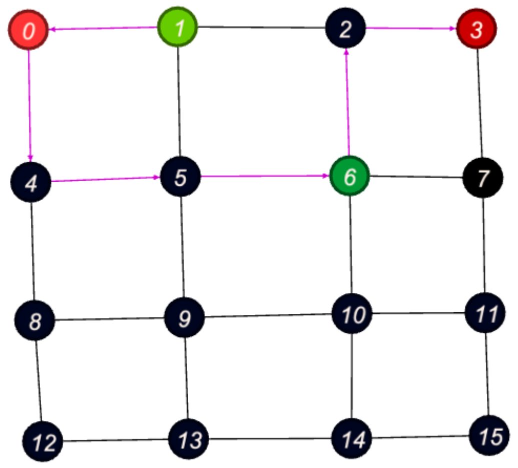


FIGURE 3.8: Overall travel path for $(p_{green}, d_{green}) \in O(r_{red}, r_{green})$

In this setting, the overall travel time for passengers r_{red} and r_{green} is 6 minutes, and 4 minutes, respectively. The overall travel time satisfy the inequality 4. For two passengers r_{red} and r_{green}, two of the four possibilities of sharing ride in their setting satisfy the inequalities in 4 and they form a feasible match.

Here we discuss the cost allocation between two passengers where they share their rides to accomplish it together. In the case that a passenger is not able to find a match and travel alone, the cost is fully paid by that passenger. When two passengers r_i and r_j share their ride, based on their order of sharing ($(p,d) \in O(r_i, r_j)$), they form a travel path that includes their both pickup and dropoff locations. In this work, we divide the trip cost equally between two passengers.

Definition 3.1.5. (Matching Cost) The matching cost defined the trip cost allocation for two passengers r_i and r_j based on their order of sharing $(p,d) \in O(r_i, r_j)$ at time t as follows:

$$M_c(r_i, r_j, (p,d), t) = \begin{cases} \frac{1}{2} C(\Im(r_i, r_j, (p,d), t), t) & if\ i \neq j \\ C(\phi_i, t) & if\ i = j \end{cases}$$

Where $C(.)$ is the trip cost function, which takes a trip path and time as inputs and returns the travel cost. It is important to note that if two passengers can share their rides on more than one order, we choose the one that minimizes the cost. The trip cost function depends on the travel distance and many other external factors, and in the next chapter, we approximate this function denoted by $\hat{C}(.)$, using historical data and machine learning method.

We assume that reducing the trip cost is the main reason for riders to share their ride, and simplification of the cost division leads to some potential match missed because of their contribution to the overall path. In 3.9, we illustrate an example of a missed potential match which not considered due to simplification in the cost function.

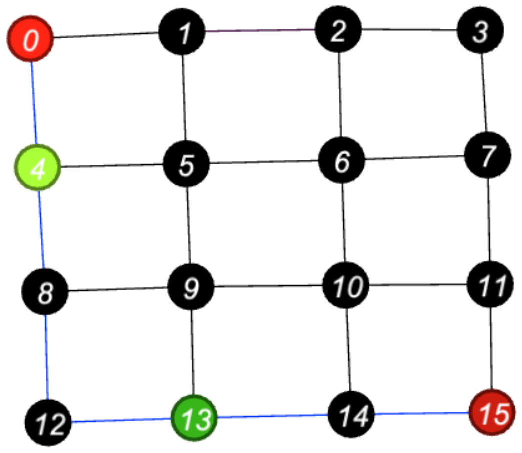


FIGURE 3.9: An example of a potential match that missed by cost simplification

In figure 3.9, r_{red} wants to go from node 0 to node 15 and r_{green} wants to go from node 4 to node 13. r_{red} and r_{green} form a feasible match for $\varepsilon = 1$, and the traveled distance reduced from 9 (in the case which each travel alone) to 6 by moving from 0 to 15 on the blue line. In this case, equally dividing the cost does not reduce the cost of r_{green}. In this example, contrary to the potential of two passengers to share their ride, they not considered due to cost constraints. This example shows us the importance of considering each passenger's contribution to the overall path in cost allocation. Here we provide a cost allocation function with better generalization. As an alternative for our cost allocation, we divide the trip cost proportionally to each passenger's contribution to the overall path. Passengers pay equally for the portion of the path where both are on the vehicle and pay for the portion where they travel alone. In the mentioned example, r_{green} is paid for half of traveling over $4 \rightarrow 8 \rightarrow 12 \rightarrow 13$, and r_{red} is paid for the other half and the cost related to remaining path that travels alone.

Suppose two passengers, r_i and r_j, forms a feasible match at time t on $(p,d) \in O(r_i, r_j)$

and the overall travel path is $\Im(r_i, r_j, (p,d), t)$. We divide the trip path in three disjoint set, $\Im(r_i, r_j, (p,d), t)_{r_i}$ is the part of the path which only r_i is on board, $\Im(r_i, r_j, (p,d), t)_{r_j}$ denotes the part of path which only r_j is on board and finally $\Im(r_i, r_j, (p,d), t)_{r_i, r_j}$ denotes the part of path which both of the passengers are on board. In this setting, r_i is going to pay:

$$C(\Im(r_i, r_j, (p,d), t)_{r_i}, t) + \frac{1}{2} C(\Im(r_i, r_j, (p,d), t)_{r_i, r_j}, t)$$

Moreover, r_j is going to pay:

$$C(\Im(r_i, r_j, (p,d), t)_{r_j}, t) + \frac{1}{2} C(\Im(r_i, r_j, (p,d), t)_{r_i, r_j}, t).$$

In the next section, the waiting time function is described in detail. When we solve an offline problem, we have the whole input in the first place, and the input does not change over time. On the other hand, when dealing with an online problem, the number of inputs streams over time, and we do not know about the upcoming events. In ridesharing, when a passenger sends his/her request for a ride, the system needs to find a match in short notice. In this work, we define the adaptive waiting time that indicates the exact moment that a passenger must start the trip and match to other passengers.

3.2 Waiting Time Function

Waiting time is the key enabler of our approach for matching. Waiting time assigns the exact time that a passenger needs to wait in the passengers pool before the matching procedure begins. In this work, the waiting time is modeled as an expectation function, and the objective is to minimize the passenger's expected trip cost. Let us suppose a passenger, $r_I = (p_I, d_I, t_I)$ who enters the system at time t_I to travel from location p_I to location d_I. For the passenger r_I, the system has the information about the pickup (p_I) and dropoff location (d_I), and request time (t_I). For passenger r_I based on inequality 2, we have $w_I + \tau_I \leqslant (1+\varepsilon)|\phi_I|$ and we can conclude $w_I \leqslant (1+\varepsilon)|\phi_I| - \tau_I$. Considering the fact that the actual trip time is greater or equal to the trip time on shortest path ($\tau_I \geqslant |\phi_I|$), we can conclude:

$$w_I \leqslant (1+\varepsilon)|\phi_I| - \tau_I \leqslant (1+\varepsilon)|\phi_I| - |\phi_I|$$

Moreover, we have the following upper bound for each passenger r_I waiting time.

$$w_I \leqslant \varepsilon|\phi_I|$$

We define the maximum possible waiting time for passenger r_I as w'_{r_I}, which is equal to $\varepsilon\phi_I$. The time interval $[t_I, t_I + w'_{r_I}]$ is discretized into one minute intervals and is denoted by $T = \{T_1, \ldots, T_k\}$ where $T_1 = t_I$ and $T_k = t_I + w'_{r_I}$.

For a waiting time μ, we define a multiset $\iota_{r_I,\mu}$ as the multiset of all requests that appears to system between t_I and $t_I + \mu$ based on the forecasted demands. It is important to note that a request could appear multiple times in $\iota_{r_I,\mu}$. Furthermore, we define the set $\upsilon_{r_I,\mu}$ as the set of unique elements in $\iota_{r_I,\mu}$. For each $r \in \iota_{r_I,\mu}$, $prob_r$ is defined as $\frac{|r \in \iota_{r_I,\mu}|}{|\iota_{r_I,\mu}|}$, where $|r \in \iota_{r_I,\mu}|$ denotes the number of requests similar to r in $\iota_{r_I,\mu}$ and $|\iota_{r_I,\mu}|$ denotes the total number of requests in $\iota_{r_I,\mu}$. To be precise, $prob_r$ calculates the probability of a request r appeared in the system between t_I and $t_I + \mu$. Using $\upsilon_{r_I,\mu}$ and $prob_r \forall r \in \upsilon_{r_I,\mu}$, we can find if r and r_I form a feasible match based on the definition 3.1.4. We split all the requests in $\upsilon_{r_I,\mu}$ to two disjoint subsets χ_f and $\chi_{\neg f}$. For a virtual request $r = (v, w, t) \in \upsilon_{r_I,\mu}$, if r_I and r are forming a feasible match at $t_I + \mu$ then we add r to χ_f. On the other hand, if r and r_I do not make a feasible match at $t_I + \mu$, we add r to $\chi_{\neg f}$.

Now, we have all the preliminaries needed to calculate the expected travel cost for the passenger r_I with waiting time μ. We define the waiting time function $w(r_I)$ as a function which aims to minimize the expected travel cost for passenger r_I as follows:

$$w(r_I) = \underset{\mu \in \{0, \ldots, w'_{r_I}\}}{\operatorname{argmin}} \ \Gamma(r_I, \mu) + \Psi(r_I, \mu) \tag{5}$$

In equation 5, our goal is to minimize the expected travel cost for passenger r_I by finding the proper waiting time.

Now, we explain the right hand side of the equation 5. The term $\Gamma(r_I, \mu)$ denotes the expected travel cost where the virtual passenger $r = (v, w, t)$ forms a feasible match to share a ride with passenger r_I at time $t_I + \mu$; defined as follows:

$$\Gamma(r_i, \mu) = \sum_{r \in \chi_f} \min_{(p,d) \in O(r_i, r)} M_c(r_i, r, (p, d), t_i + \mu) prob_r \tag{6}$$

where $\underset{(p,d) \in O(r_I, r)}{\operatorname{argmin}} M_c(r_I, r, (p, d), t_I + \mu)$ is the cost allocation for their trip and $prob_r$ is the probability of appearance of r in the system . The term $\Psi(r_I, \mu)$ denotes the expected travel cost where for the virtual passenger $r = (v, w, t)$ it is not feasible to share a ride with passenger r_I at time $t_I + \mu$; defined as follows:

$$\Psi(r_i, \mu) = M_c(r_i, r_i, (p_i, d_i), t_i + \mu) \sum_{r \in \chi_{\neg f}} prob_r \tag{7}$$

where $M_c(r_I, r_I, (p_I, d_I), t_I + \mu)$ is the travel cost for passenger r_I when traveling alone. In equation 5 the optimal waiting time is calculated on a discrete time (one minute) scale based on the probability of a passenger $r = (v, w, t)$ that appears in the system between t_I and $t_I + \mu$. For each passenger r who has sent a request for service we can calculate the optimal waiting time by equation 5 then update the passenger's request $r = (p, d, t)$ to $r = (p, d, t, w)$. At the time instance $t + w$, matching procedure starts for passenger r to share a ride with passengers in Π_{t+w}.

3.3 Passengers Graph

After adding the waiting time, the passenger r_I can be expressed as $r_I = (p_I, d_I, t_I, w_I)$ and suppose the current time is T_c, obviously for each passenger r_i in Π_{T_c}, $t_i \leqslant T_c \leqslant t_i + w_i$, we define the set of leaving passengers at time T_c as the set of passengers when the waiting time has elapsed and ready to match; formally defined as follows:

Definition 3.3.1. (Leaving Passengers) The leaving passengers are the set of all passengers $r_i \in \Pi_{T_c}$, which are leaving the system concerning their waiting and request time.

$$L_{T_c} = \{r_i \in \Pi_{T_c} : t_i + w_i = T_c\}$$

Note that for each passenger r_i in Π_{T_c}, $t_i + w_i \geqslant T_c$, furthermore the leaving passenger is the set of all passengers in Π_{T_c} which $t_i + w_i = T_c$. For any time instance T_c, if the leaving candidates set is not empty, then the matching procedure starts. Before introducing the matching procedure, the passenger graph must be defined. For defining the passenger graph G_p at time T_c, the set of vertices $V(G_p)$ and the set of edges $E(G_p)$ must be defined.

Let suppose $|L_{T_c}| = k$ and $|\Pi_{T_c}| = h$ and $\Pi_{T_c} = \{r_1, \ldots, r_h\}$ and without loss of generality, suppose $L_{T_c} = \{r_1, r_2, \ldots, r_k\}$. For the passenger graph G_p the vertices are divided into two parts, L and P, where the vertices in $L = \{v_1, \ldots, v_k\}$ and vertices in $P = \{v'_1, v'_2, \ldots, v'_h\}$. The set P in G_p is an independent set, in other word, there is no edge between the vertices in P. All the edges are between vertices in L and also between vertices in P and L. For $i \in \{1, 2, \ldots, k\}$ there is an edge between v_i, v'_i with weight $w(r_i, r_i) = M_c(r_i, r_i, (p_i, d_i), T_c)$, these set of edges are added as passengers can travel alone. For $i \in \{1, 2, \ldots, k\}$ and $j \in \{k+1, \ldots, h\}$, an edge exists between v_i and v'_j with weight equal to $w(r_i, r_j) = \underset{(p,d) \in O(r_i, r_j)}{\operatorname{argmin}} M_c(r_i, r_j, (p, d), T_c)$, if passenger r_i and passenger r_j form a feasible match and $w(r_i, r_j) \leqslant M_c(r_i, r_i, (p_i, d_i), T_c)$ and $w(r_i, r_j) \leqslant M_c(r_j, r_j, (p_j, d_j), T_c)$. Also, for $i, j \in \{1, 2, \ldots, k\}$ there is an edge between v_i and v_j with weight equal to $w(r_i, r_j) = \underset{(p,d) \in O(r_i, r_j)}{\operatorname{argmin}} M_c(r_i, r_j, (p, d), T_c)$ if two leaving candidates r_i and r_j form a feasible match and $w(r_i, r_j) \leqslant M_c(r_i, r_i, (p_i, d_i), T_c)$ and $w(r_i, r_j) \leqslant M_c(r_j, r_j, (p_j, d_j), T_c)$.

Here we use an example to explain how to create G_p for leaving passengers at time instance T_0. Suppose 15 passengers exists in Π_{T_0} and 5 passengers are in L_{T_0}. Then let's assume the requests $\{r_1, \ldots, r_{15}\}$ are in the passengers pool and $\{r_1, \ldots, r_5\}$ is the set of leaving passengers. As described above the graph G_p has 20 vertices, $L = \{v_1, \ldots, v_5\}$ indicates the vertices which represent the leaving passengers and each edge in G_p has at least one end in L and the set $P = \{v'_1, \ldots, v'_{15}\}$ which represents the passengers pool. First, we add the edges between $(v_i, v'_i) \forall i \in \{1, 2, ..., 5\}$. These edges are responsible to the case where passenger r_i should travel alone. The vertices and the edges are shown in figure 3.10.

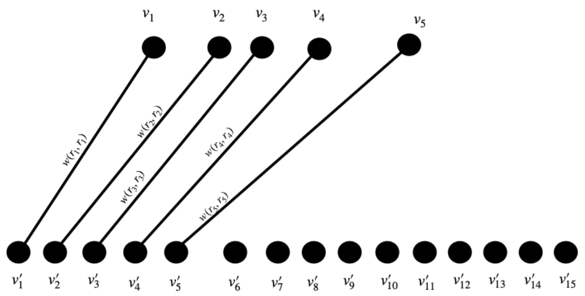


FIGURE 3.10: Edges between identical vertices in passengers graph

Then, it is time to add edges between the passengers in L, for two vertices $v_i, v_j \in L$ we add the edge $e(v_i, v_j)$ with weight of $w(r_i, r_j)$ if r_i and r_j are able to share a ride while satisfying the constraints. In this example, let's assume passengers r_1 and r_2 are able to share a ride and passengers r_4 and r_5 are also able to share another ride. We add an edge between v_1, v_2 with weight equals to $w(r_1, r_2)$ and add an edge between v_4, v_5 with weight equals to $w(r_4, r_5)$; shown in figure 3.11.

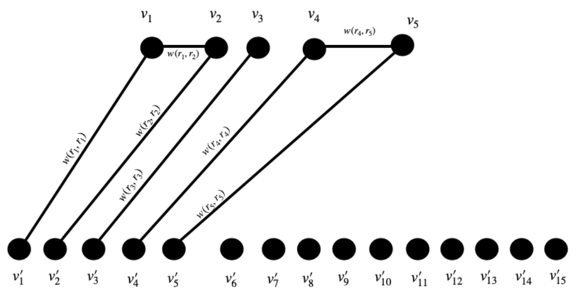


FIGURE 3.11: Identical and inner L edges added to G_P

For two passengers $r_i \in L_{T_0}$ and $r_j \in \Pi_{T_0}/L_{T_0}$, there is an edge between the representing vertices, namely, v_i and v'_j if the passengers form a feasible match. In this example suppose r_3 and r_7 are able to share a ride and also r_4 is able to share a ride with r_8 and r_10. For covering this possibility for sharing a ride, the edge set $\{e(r_3, r_7), (r_4, r_8), (r_4, r_9)\}$ are added to the graph with their weight equal to cost associated with the shared trip, and the graph G_p has finalized by adding the vertices and edges. The passenger graph is shown in figure 3.12.

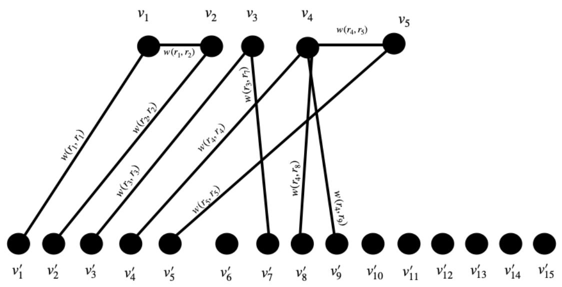


FIGURE 3.12: The passengers graph

The graph G_p is used to match the passengers. In graph theory, a *matching* on a graph is a set of edges with no common vertex. For matching M which is a set of edges in G_p, the vertices that are endpoints of edges in M called saturated. In this work we want to find a matching which saturates the vertices in L. For the graph which we established in the above example, $M_O = \{e(v_1, v'_1), e(v_2, v'_2), e(v_3, v'_3), e(v_4, v'_4), e(v_5, v'_5)\}$ is a matching set which is saturated 10 vertices $v_1, v_2, v_3, v_4, v_5, v'_1, v'_2, v'_3, v'_4, v'_5$. As a result of matching, each passenger in L_{T_0} is traveling alone and the cost associate with this matching is $C_{M_O} = w(r_1, r_1) + w(r_2, r_2) + w(r_3, r_3) + w(r_4, r_4) + w(r_5, r_5)$. Furthermore, for a matching M the cost of matching is equal to sum of the weights of selected edges. A *maximal matching* is a matching M of a graph G_p which adding any edge to M causes M to not be a matching set. As an example, M_O is a maximal matching in the above mentioned example.

Theorem 1. Any maximal matching in G_p saturates all the vertices in L.

Proof. Suppose M is a maximal matching and not saturated all the vertices in L. Then, there is a $v \in L$, which M does not saturate v. The identical vertex to v, $v' \in P$ is also not saturated by M because the degree of v' is one, and its only neighbor is v. By adding the edge $e(v, v')$ to M, M is still a matching and bigger than M, then it is a contradiction. $\square$

By theorem 1, every maximal matching saturates the vertices in $L \subset V(G_p)$, and we want to find the one which minimizes the total weight of edges in matching in order to minimize the total cost for leaving passengers. It is also important that all the edges are at least one endpoint in set L, and in other words, P is an independent set.

3.4 Matching Procedure

The procedure for creating the passenger graph G_p is explained in the last section, and theorem 1 shows that every maximal matching saturates all the vertices in L. One of the advantages of ridesharing is that sharing the ride and related expenses could reduce the travel costs for passengers, and it is considered one of the main advantages of ridesharing over the taxi or ride-hailing. We aim to encourage passengers to share their ride in order to achieve a better overall system performance. To encourage passengers, we propose two mechanisms in order to provide cheap and quality trips. From the convenience perspective, we defined ε detour over the shortest path to ensure the trip duration did not exceed from a threshold. Second, by finding the minimal weight maximal matching, we want to minimize the trips' overall cost.

In this section, we discuss two approaches in finding the minimal weight maximal matching in graph G_p. In the first approach, the minimal weight maximal matching formulated as an integer linear programming (ILP) and solved to find the exact solution. One difficulty arises with this approach because when the number of vertices and edges in the graph G_p increases, solving the ILP becomes computationally intractable, and it is not practical to find the optimal solution in a short notice to handle the numerous requests. To overcome this drawback in finding the exact solution, we propose a greedy algorithm that can find a maximal matching in graph G_p.

3.4.1 Optimization Approach

There are different maximal matchings for G_p, and we are interested in one with the minimum sum of the edges' weight. In other words, our goal is to find minimum weight maximal matching [92]. Minimum weight maximal matching represents a maximal matching between passengers with the minimum total cost for the trips.

For introducing the optimization formulation of the minimum weight maximal matching, first we define preliminaries. Let $G_p = (V, E)$ be a graph with the vertex set $V = L \cup P$

where $L = \{v_1, \dots, v_k\}$ is the set of leaving passenger and $P = \{v'_1, \dots, v'_h\}$ is the set of all the passengers in passengers pool while edge set is $E = E_L + E_P$ where $E_L = \bigcup e(v_i, v_j)$ and $E_P = \bigcup e(v_i, v'_j)$. As discussed in passenger graph section, each edge in E have at least one end in L. For a vertex $v \in V$, $N(v)$ denotes the neighbors of v, in other words, $x \in N(v)$ if $e(v,x) \in E$. For each $e(v,w) \in E$ where $v,w \in V$ and c_{vw} denotes the weight of the edge.

We use the integer linear programming (ILP) formulation for minimum weight maximal matching as introduced in [92]. We define x_{v_i,v_j} and x_{v_i,v'_j} as binary variables and $x_{v_i,v_j} = 1$ if the edge $e(v_i, v_j)$ has been selected by optimal minimum weight maximal matching and $x_{v_i,v'_j} = 1$ if the edge $e(v_i, v'_j)$ has selected by optimal minimum weight maximal matching. And also define the binary variable y_{v_i} for vertices in L and $y_{v'_j}$ for vertices in P. $y_{v_i} = 1$ if vertex $v_i \in L$ is saturated by optimal minimum weight maximal matching and $y_{v'_j} = 1$ if vertex $v_j \in P$ is saturated by optimal minimum weight maximal matching.

$$\begin{array}{llll}
\text{minimize} & \displaystyle\sum_{e(v_i,v'_j)\in E} c_{v_iv'_j} x_{v_i,v'_j} + \sum_{e(v_i,v_j)\in E} c_{v_iv_j} x_{v_i,v_j} & & (a) \\
\text{subject to:} & \displaystyle\sum_{j\in N(v'_i)} x_{v'_i,j} = y_{v'_i}, & \forall v'_i \in P & (b) \\
 & \displaystyle\sum_{j\in N(v_i)} x_{v_i,j} = y_{v_i}, & \forall v_i \in L & (c) \\
 & y_{v_i} + y_{v'_j} - x_{v_i,v'_j} \geqslant 1, & \forall e(v_i, v'_j) \in E & (d) \\
 & y_{v_i} + y_{v_j} - x_{v_i,v_j} \geqslant 1, & \forall e(v_i, v_j) \in E & (e) \\
 & x_{v_i,v_j} \in \{0,1\} & \forall e(v_i, v_j) \in E & (f) \\
 & x_{v_i,v'_j} \in \{0,1\} & \forall e(v_i, v'_j) \in E & (g) \\
 & y_{v_i} \in \{0,1\} & \forall v_i \in L & (h) \\
 & y_{v'_i} \in \{0,1\} & \forall v'_i \in P & (i)
\end{array}$$

The objective function (a) minimizes the total weight of the selected edges. Constraints (b) enforce the condition that vertex $v'_i \in P$ is saturated ($y_{v'_i} = 1$) if exactly one edge between v'_i and its neighbor ($N(v'_i)$) is selected in the matching, in other words, it makes sure the set of selected edges form a matching. Constraints (c) enforce the condition that vertex $v_i \in L$ is saturated ($y_{v_i} = 1$) if exactly one edge between v_i and its neighbor ($N(v_i)$) is selected in the matching. In this way it makes sure the selected edges will form a matching. Constraints (d) induce that if $x_{v_i,v'_j} = 1$ then y_{v_i}, and $y_{v'_j}$ must be equal to 1, in other word by selecting $e(v_i, v'_j)$ both the end-point vertices are saturated. Otherwise, if $x_{v_i,v'_j} = 1$, then at least one of the $y_{v'_j}$ or y_{v_i} has to be saturated to ensure maximality of the matching. Constraints (f,g,h,i) enforce the added variable to take binary values.

The number of binary variables for solving the graph G_p is equal to $|V| + |E|$. Solving the 3.4.1 optimization problem to find the exact solution is a time-consuming task, and in our experiment, it takes up to 500 seconds to solve the problem with 160 vertices. The nature of our task, which is matching passengers to each other, should be able to run in a short time for graphs with hundreds of nodes. Designing an algorithm that is able to find the exact solution for minimum weight maximal matching is not tractable because our problem is $\mathcal{NP}$, and subsequently, finding the exact solution is exponential.

3.4.2 Greedy Approach

As discussed in the previous part of this section, **Minimum Weight Maximal Matching** is an $\mathcal{NP}$-problem, and finding the exact solution in polynomial time is impossible. Furthermore, our experiments show the ILP approach can solve the problem with small size, and it is time-consuming to solve the optimization problem 3.4.1 for large and dense graphs. To overcome this problem and find a good match for the passenger in short notice, we applied a greedy approach for choosing the edge and creating a matching for graph G_p.

Our goal is to minimize the cost of matching in short notice, and we do it by taking a greedy approach. First, we define an empty set M, which will expand during the greedy approach and form a matching at the end. We start by sorting the edges by their cost, picking the one with the lowest cost, adding it to M, and removing the selected edge and end-point vertices from G_p. At each iteration at least one of the vertices in L is removed, and after maximum k iteration the matching is formed.

Algorithm 1: Greedy matching

```
1  Input: A graph $G_p$
2  Output: A matching set M
3  $M \leftarrow \{\}$;
4  Sort $E$;
5  while $L$ is not empty do
6      select $e \in E$ with minimum cost;
7      $M \leftarrow M \cup e$;
8      remove $e$ and adjunct vertices from $G_p$;
9  end
10 return $M$;
```

The greedy approach for matching the passengers in G_p is provided in algorithm 1. In the next theorem, we show that for any graph G_p, the greedy approach forms a maximal matching and saturate all the vertices in L.

Theorem 2. The algorithm 1 for any graph G_p forms a maximal matching and saturate all the vertices in L.

Proof. First, we show that M forms a matching. We prove it by contradiction. Let us suppose M is not matching; then there is at last one vertex that saturates twice and name it v. During the selecting edges by algorithm 1 suppose e be the edge which adjunct to v and first appears in M. After adding e to M, v removed from G_p, and it is not possible to saturate twice, so the set M forms a matching. For the maximality of M, let us suppose M is not maximal, then there exists some $e \in E$ where $M \cup \{e\}$ forms a bigger matching, on the other hand, M saturates all the vertices in L and all the edges are removed from G_p; thus e does not exist, and M is maximal. □

In the end, we compute the complexity of the algorithm. The sorting part complexity is $O(|E|log|E|)$, and after sorting the edges in E based on their weights, we move over the sorted edge sets and add edges to M. The complexity of the greedy algorithm is also $O(|E|log|E|)$. At the beginning of the ridesharing simulation, we provide an algorithm to cover all the procedures we explain and formalize here. This includes the waiting time function, the passengers graph, and the matching procedure.

Chapter 4

Data-Driven Function Approximation

In this chapter, data-driven function approximation is used to approximate the trip duration function ($\hat{\phi}$ and trip cost function ($\hat{C}$) as the essential components in the execution of the ridesharing simulation. New York City taxi and limousine commission publicly released the taxi trip record data between 2009 and 2019 [93]. The original dataset includes more than 2 billion trips, and a portion of the data is used in our function approximation technique to train and evaluate the regression models that take a trip and return trip duration and trip cost. Besides that, the request in simulation comes from this dataset.

Here we first explain our learning approach for predicting the duration and the cost functions. Supervised learning, which is one of the main fields in Artificial Intelligence, can be divided into two subgroups, namely, regression and classification. In classification, the number of possible outputs is fixed and known. For example, in image classification, the number of categories is known. On the other hand, in regression, the prediction space is continuous, and the function can return infinite value. For both the cost function and duration function, regression is used to learn a projection of requests to $\mathbb{R}$. Next, we explore the taxi data to find the features that contribute to decision making and add them to the data instances to train a better model with less loss. Finally, we provide the information regarding the training procedure and also the evaluation results.

4.1 Learning Approach for Regression

4.1.1 Tree Based Methods

Tree based methods are one of the simple but powerful approaches in learning from data. The tree based method partitions the feature space into regions and afterward fits a simple model in each one. Let suppose our data consists of d feature and a label, for each of N observation: (x_i, y_i) for $i = 1, 2, ..., N$, with $x_i = (x_{i1}, x_{i2}, ..., x_{id})$. The learning algorithm needs to decide on the important variable for splitting and topology of the tree. Suppose we have M regions, namely $R_1, R_2, ..., R_M$, and the response is a constant r_m in each region:

$$f(x) = \sum_{m=1}^{M} r_m I(x \in R_m). \tag{8}$$

If the objective be the minimization of the sum of squares $\sum(y_i - f(x_i))^2$, then the optimal $\hat{r}_m$ is the average of y_i in region R_m for $m \in \{1, \ldots, M\}$:

$$\hat{r}_m = avg(y_i | x_i \in R_m)$$

Finding the optimal partitioning in terms of the minimum sum of squares is computationally infeasible [43] and to be exact, [50] proved the problem of constructing optimal binary tree is $\mathcal{NP}$ by reducing it to EC3. To overcome the computational infeasibility, a greedy approach is used to define the pair of half-planes on splitting variable j and split point s, defined as follows:

$$R_1(j,s) = \{x | x_j \leqslant s\} \; and \; R_2(j,s) = \{x | x_j > s\}. \tag{9}$$

for solving equation 9, we need to minimize squared error in this half-plane:

$$\min_{j,s} \left[\min_{r_1} \sum_{x_i \in R_1(j,s)} (y_i - r_1)^2 + \min_{r_2} \sum_{x_i \in R_2(j,s)} (y_i - r_2)^2 \right]. \tag{10}$$

By fixing the value of the j and s, the optimal solution for the inner minimization is solved by

$$\hat{r}_1 = avg(y_i | x_i \in R_1(j,s)) \; and \; \hat{r}_2 = avg(y_i | x_i \in R_2(j,s)). \tag{11}$$

By observing the data we can determine the s value for each variable and subsequently determine the optimal pair (j,s) [43]. In this way, we divide the data into two regions, namely, R_1, R_2. By repeating this procedure on regions, we can expand the number of regions for prediction. The other important factor in tree based regression is the size of the tree. One approach is to grow a large tree T_I and prune it by a cost-complexity pruning approach as follows:

A pruned subtree $T \subset T_I$ is obtained by pruning any number of T_I non-terminal nodes. let $|T|$ be the number of region (terminal nodes) in subtree T, N_m be the number of point in R_m, $\hat{r}_m = avg(y_i|x_i \in R_m)$ and $Q_m(T) = avg((y_i - \hat{r}_m)^2|x_i \in R_m)$, then the cost complexity function is defined as:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|. \tag{12}$$

In the equation 12, $\alpha|T|$ is the regularization term and defined as a penalty on the model complexity. let $\hat{R}(T)$ be the empirical risk of T $(\sum_{m=1}^{|T|} N_m Q_m(T))$, then we can rewrite equation 12 as follows:

$$C_\alpha(T) = \hat{R}(T) + \alpha|T|. \tag{13}$$

Cost complexity pruning has 2 steps: First, for each α, $T_\alpha \subseteq T_I$ is a subtree which minimizes the $C_\alpha(T)$. The second step is to find the right choice of α using cross validation. To find T_α, we successively collapse the internal node that produces the smallest per node increase in $\hat{R}(T)$ and continue till we have a single-node (root) tree. Suppose $\chi = \{T_I \supset T_1 \supset ... \supset T_K\}$ where K is the number of internal nodes in T_I. [15] proved that the optimal answer is in χ.

$$\left\{ \underset{T \subset T_I}{\operatorname{argmin}} C_\alpha(T) | \alpha \geqslant 0 \right\} \subset \chi$$

.

4.1.2 Boosting Methods

Boosting is one of the most powerful approaches for learning from data. [18] by an extensive empirical comparison of 10 different classifiers, showed that boosted trees are the best in terms of misclassification error and producing well-calibrated probabilities. Boosting first was introduced in computational learning theory in [34], where the focus was on binary classification. [14] showed that boosting can be interpreted as gradient descent in function space, and [35] extended the boosting approach to handle a variety of loss functions, including regression. The boosting works by combining many weak classifiers/regressors to produce a powerful classifier/regressor. The algorithms work by applying the base learner sequentially to a weighted version of the data, where more weight is given to misclassified examples by earlier learners to boost the current tree in classifying them correctly. Here we explain the boosting method in detail based on [43]:

Regression tree partitions the feature space into disjoint regions $R_i, i = 1, 2, ..., M$. A constant r_i is assigned to each region and the prediction is made by the following role:

$$x \in R_i \implies f(x) = r_i$$

A tree can be expressed as follows:

$$T(x;\Theta) = \sum_{i=1}^{M} r_i I(x \in R_i),$$

With parameters $\Theta = \{R_i, r_i\}_1^M$. The parameters are modified by minimizing the empirical risk as follows:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{j=1}^{M} \sum_{x_i \in R_j} L(y_i, r_j) \tag{14}$$

Where $L(.)$ is the loss function and in our case root-mean-square error (RMSE) used as loss function. Solving the equation mentioned above is discussed in the beginning of this section and here we focus on boosted trees. The boosted trees model is the sum of a series of K trees and defined as follows:

$$f_K(x) = \sum_{k=1}^{K} T(x;\Theta_k). \tag{15}$$

Here we have K trees instead of one, and the parameters for optimization are $\{\Theta_k\}_1^K$. The optimization problem is expanded to the following form:

$$\hat{\Theta}_k = \underset{\Theta_k}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, f_{k-1}(x_i) + T(x_i, \Theta_k)). \tag{16}$$

Numerical optimization is used to solve the optimization problem 16. Suppose the loss for predicting y using $f(x)$ on the training data is:

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)).$$

The goal is to minimize $L(f)$ with respect to f, where f is the sum of trees as explained in equation 15. Minimizing $L(f)$ can be solved as an optimization problem in the following form:

$$\hat{f} = \underset{f}{\operatorname{argmin}} L(f), \tag{17}$$

where $f \in \mathbb{R}^N$ are the values of the approximating function $f(x_i)$ for each of the training points x_i.

$$f = \{f(x_1), f(x_2), ..., f(x_N)\}.$$

At step k, we can get g_k as the gradient of $L(f)$ evaluated at $f = f_{k-1}$

$$g_{ik} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]. \tag{18}$$

The parameter is updated using the gradient by the following rule:

$$f_k = f_{k-1} - \rho_k g_k \tag{19}$$

where ρ_k is the step length, chosen by:

$$\rho_k = \underset{\rho}{\operatorname{argmin}} L(f_{k-1} - \rho g_k).$$

We can modify the algorithm by fitting a weak learner to approximate the negative gradient, and reach the following update for the prediction parameters:

$$r_k = \underset{r}{\operatorname{argmin}} \sum_{i=1}^{N} (-g_{ik} - T(x_i, \Theta))^2 \tag{20}$$

the following algorithm summarizes the gradient boosting method.

Algorithm 2: Gradient boosting

```
1 initialize $f_0(x) = \underset{r}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, r)$;
2 for $k$ = 1 to $K$ do
3     for $i = 1, 2, ..., N$ do
          $g_{ik} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{k-1}}$
4     end
      Fits a regression tree to $g_{ik}$ values and create the terminal regions $R_{jk}$, for
      $j = 1, 2, ..., M_k$.
      for $j = 1, 2, ..., M_k$ do
          $r_{jk} = \underset{r}{\operatorname{argmin}} \sum_{x_i \in R_{jk}} L(y_i, f_{k-1}(x_i) + r)$
      end
      Update $f_k(x) = f_{k-1}(x) + \sum_{j=1}^{M_k} r_{jk} I(x \in R_{jk})$
5 end
6 return $\hat{f}(x) = f_K(x)$.
```

There are many different implementations on the gradient boosting method, which provides an end-to-end tree boosting framework to learn from the data. **XGBoost** is one of them by developing a novel method for handling the sparse data, and a theoretically justified weighted quantile sketch for approximate learning [21]. The other work we used in this work for function approximation is called **LightGBM** developed by Microsoft research [53]. Gradient boosting machines need to scan the whole data and each feature to estimate the information gain of all the possible split points. Therefore, the computational complexity depends on the number of training examples N and the number of features D, and it is time-consuming when we face big data. They proposed two novel methods to tackle the mentioned problem [53].

Gradient-based One-Side Sampling used a gradient method to find the sample, which contributes the most to the information gain. In this way, at each step, they exclude a significant proportion of data with a small gradient and estimate the information gain for the remaining part of the data. They proved that the gradient-based sampling, as described above, leads to more accurate information gain than uniform sampling [53]. *Exclusive Feature Bundling* designed to bundle features into a single feature by reducing the problem into graph coloring problem, which is a well-known $\mathcal{NP}$-hard problem and uses a greedy approach to solve it [53].

The gradient boosting approach is used as the learning procedure for both the **trip cost prediction**, and **trip duration prediction**. However, one of the most important parts of the model design remains. Most of the machine learning algorithms rely on the parameters, and these parameters are crucial because the training procedure depends on them and is controlled by them. For the gradient boosting trees, the parameters include:

- Learning Rate
- Depth
- Feature Fraction
- Bagging Fraction
- Frequency for Bagging
- λ_{L_1}
- λ_{L_2}

Each of the mentioned parameters had an essential role in the performance of the trained model. Here we describe the primary role of the hyperparameters, and afterward, we discuss how to find the optimal hyperparameters. The learning rate is the most crucial feature in the gradient-based optimization approach and scales the magnitude of the parameter with respect to the gradient. A large learning rate could cause divergent behavior, and a small one could require too many steps to find the optimal parameters. Depth indicates the maximum depth each tree could be grown, especially in small data sets with a limited number of features; it can avoid over-fitting by increasing the number of samples in terminal nodes. Feature fraction could be between 0 and 1, and for each tree, a randomly selected portion of the features is used for training. Bagging fraction works the same as the feature fraction and selects a portion of the data for training the tree. Both the feature fraction and bagging fraction are used to avoid over-fitting and speed up the training procedure. The frequency for bagging indicates the frequency that bagging is used to reduce the number of training elements. λ_{L_1} and λ_{L_2} are the regularization coefficients which control the over-fitting by limiting the search space and force the objective function to consider the weights of the model.

For finding the optimal value for the mentioned hyperparameters, we have used Bayesian optimization [87]. Bayesian optimization assumes that the unknown function is sampled from a Gaussian process and form a posterior distribution for this function as observing the loss of the learning algorithm with different hyperparameters [87]. The hyperparameters are adjusted by optimizing the expected improvement [73] or using the Gaussian process upper confidence bound [89].

4.2 Exploratory Data Analysis on The NYC Trip Data

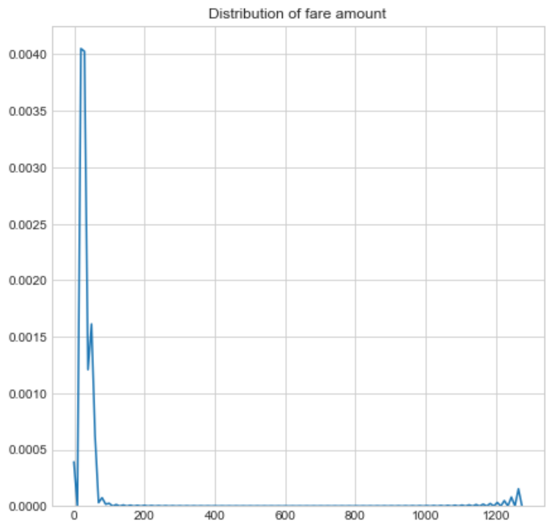
In this section, we provide the exploratory data analysis on the NYC yellow-cab data. We explore the NYC yellow-cab data to add extra features to improve the learning procedure and achieve better performance on the evaluation phase. Travel duration and travel cost are two crucial variables in our method, and a good approximation can assist our procedure to make ridesharing convenient. To train powerful regression functions, first, we extensively take a look into the data and add some features to enrich the data to increase the possibility of accurate prediction with low error. Then, gradient boosting trees are used to learn predictive models on data with high variation.

First, let us take a look into the data to have a better understanding of the trip cost and trip duration prediction. For the duration and cost, we sampled around 8 million trips from the first six months of the years between 2009 to 2015, where each instance contains the following features.

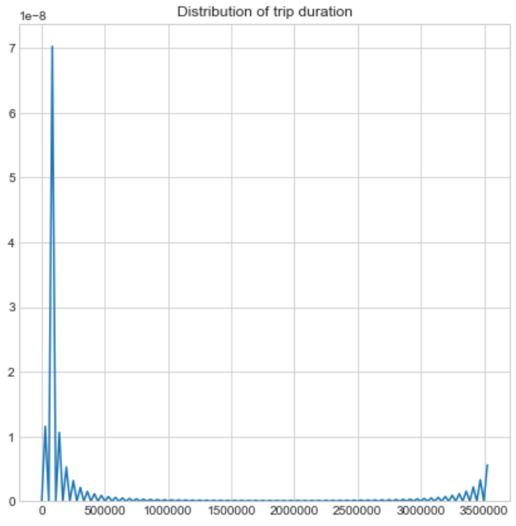
- **id** - The id is a unique identifier for the trip.
- **vendor id** - The vendor id is a code indicating the provider of the service(could be one or two).
- **pickup datetime** - Indicates the date and time that the trip started.
- **dropoff datetime** - Indicates the date and time that the trip ended.
- **passenger count** - The number of passengers in the trip.
- **pickup longitude** - The longitude where the trip started.
- **pickup latitude** - The latitude where the trip started.
- **dropoff longitude** - The longitude where the trip ended.
- **dropoff latitude** - The latitude where the trip ended.
- **store and fwd flag** - This flag indicates whether the trip record was held in vehicle memory before sending it to the vendor because the vehicle did not have a connection to the server.
- **trip duration** - The trip duration in seconds.
- **trip cost** - The trip cost in USD.

The row data consists of twelve columns. The trip cost and trip duration are the target value that we want to learn to predict for the upcoming requests. So, in the beginning, we have ten features that expand in this section and generate more features to improve the prediction results. From the features, we remove **dropoff datetime**, which carries the trip duration information. The spatial features, including **pickup longitude, pickup latitude, dropoff longitude, dropoff latitude**, indicate the pickup and dropoff location for a trip. The **pickup datetime** includes the pickup date and time and is the only temporal feature.

We start by plotting the target value (trip duration and trip cost) to visualize their distribution.

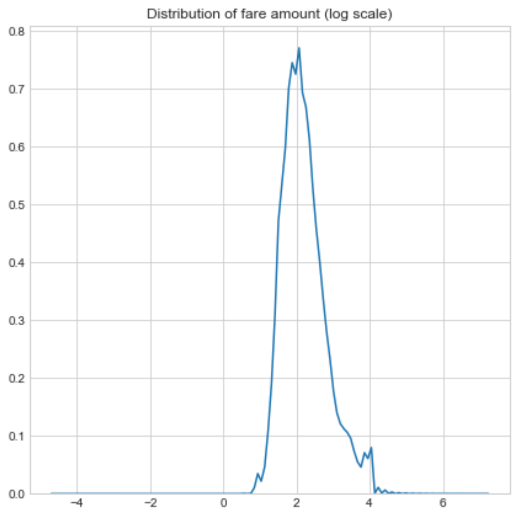


(A) Trip cost distribution

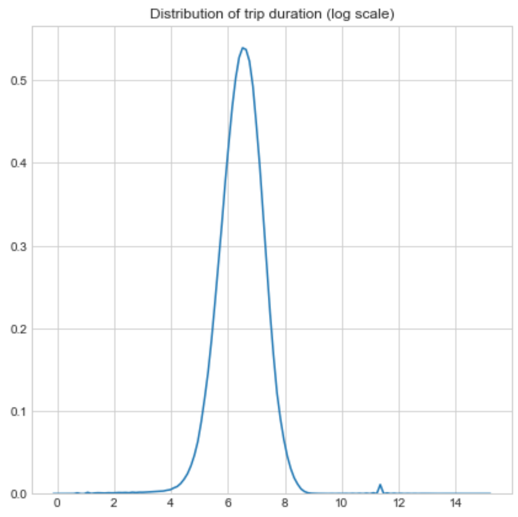


(B) Trip duration distribution

As we can see in the figures 4.1b and 4.1a, both the cost and duration for trips do not follow a normal distribution, and we plot the log of target values to see their distributions on log transform.



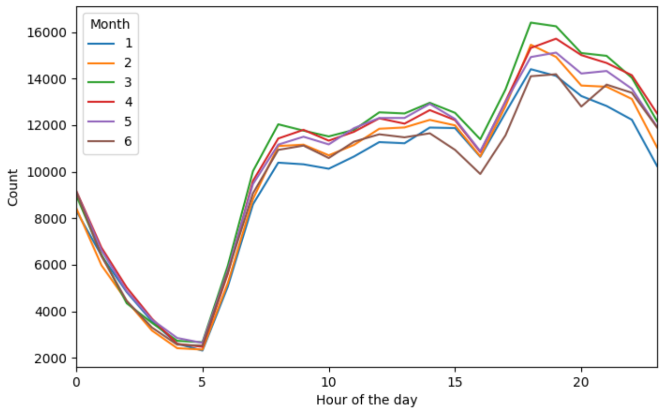
(A) Trip cost distribution (log scale)



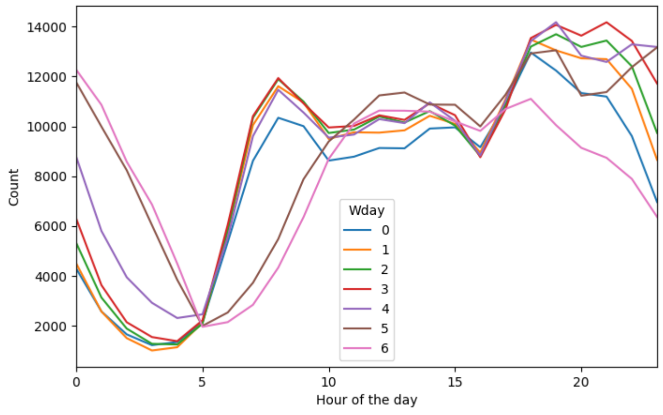
(B) Trip duration distribution (log scale)

As we observe in 4.2a and 4.2b, the cost and duration distributions show a normal distribution, and we use the log scale of the target values for prediction. After analyzing the target values and plot their distribution, temporal features explored to extract features.

The goal of adding features is to achieve a better performance in the learning procedure. We break the **pickup datetime** feature to **day of the week**, and **month**, **hour**. Here we observe the relationship between expanded features and target values.



(A) Mean number of requests over 24 hours for different months



(B) Mean number of requests over 24 hours for different weekdays

In figure 4.3a, the mean number of requests plotted for six months over 24 hours. As we can see, the mean number of requests changes significantly in a day. For example, 2000 requests came on average at 5, and at 18, the number of requests varies between 14000 and 16000 for different months. In figure 4.3b, the mean number of requests plotted for weekdays over 24 hours, and the observations are as follows:

1. The weekend shows different patterns.

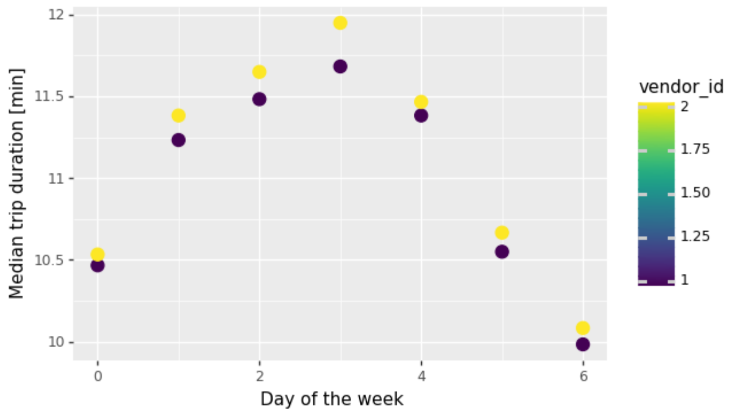
    - The number of requests between 00 AM and 05 AM is relatively higher.

- During the morning, between 5 AM to 10 AM, the number of requests is relatively lower.

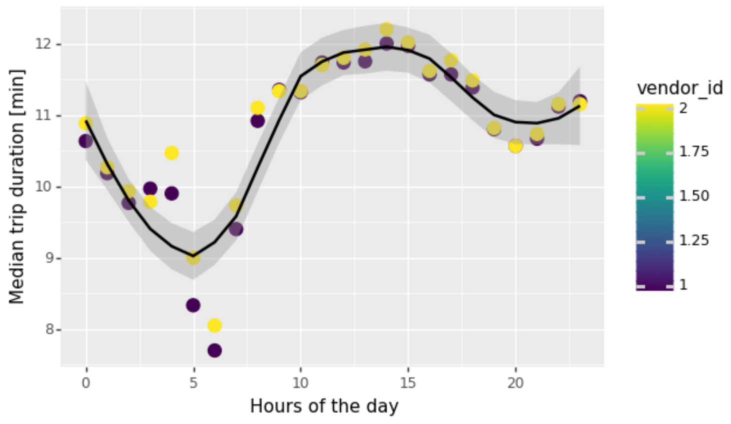
2. The request pattern in weekdays.

   - We observe two peaks in the early morning around 9 AM and 6 PM (at the beginning and end of the working hours).
   - The number of requests, in the second half of the day, is higher than the first half.

The number of requests can represent the city's transportation situation. For instance, during peak hours, the number of requests is relatively higher, and traffic congestion is also more massive due to the higher number of vehicles on the streets. To observe the relation between time and trip duration, we calculate the median of trip duration in different settings.



(A) Median trip duration over week days



(B) Median trip duration over day

As we can see, the median trip duration varies significantly over the day. For example, between 3 to 5 in the morning, we have the shortest trips, and it is related to traffic and the fact that the average travel speed during these hours is much higher than rush hour. During the rush hour from 9 to 18, the median trip duration is much higher due to traffic in the city network. The median trip for different days shows the same pattern. During the weekdays, the median trip duration is higher than the weekend. These observations suggest that temporal information plays a vital role in the trip duration and consequently trip cost. To investigate more in the relation between the request time and trip duration, we used Google map API to have google estimation for trip duration over different hours between two fixed locations in New York City. The first trip is from Columbia University to New York University with a 12-kilometer path, and the second trip is from Microsoft office to Google office with a 2.7-kilometer trip distance.

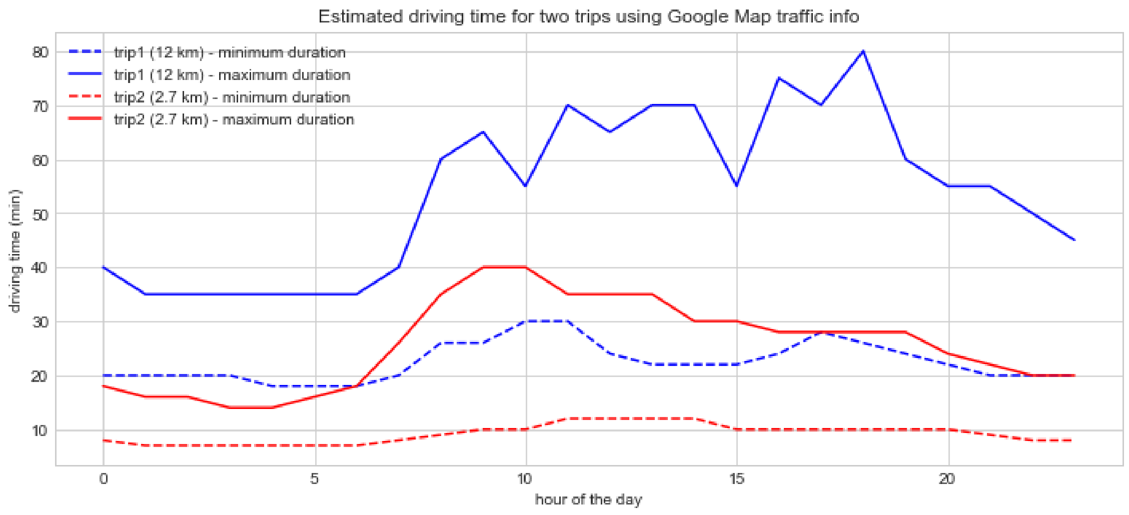


FIGURE 4.5: The expected travel duration for two trips in Manhattan

For the shorter trip (Microsoft office to Google office), the minimum trip duration is almost fixed and varies between 8 and 12 minutes. The maximum trip duration varies from 15 to 40 minutes, and we can observe the same thing for the trip from Columbia University to NYU. In the second case, the minimum trip duration varies between 18 and 30 minutes, and the maximum trip duration varies between 35 and 80 minutes. Based on our observation, the most important factor which leads to a large variation in trip duration is traffic, and traffic is a consequence of extensive usage of the roads by people.

Next, we investigate the relation between cost and distance over time. We divide a day into 24 hours and use linear regression to fit a line on the data points to observe the correlation between cost and distance, and we analyze our observation here.

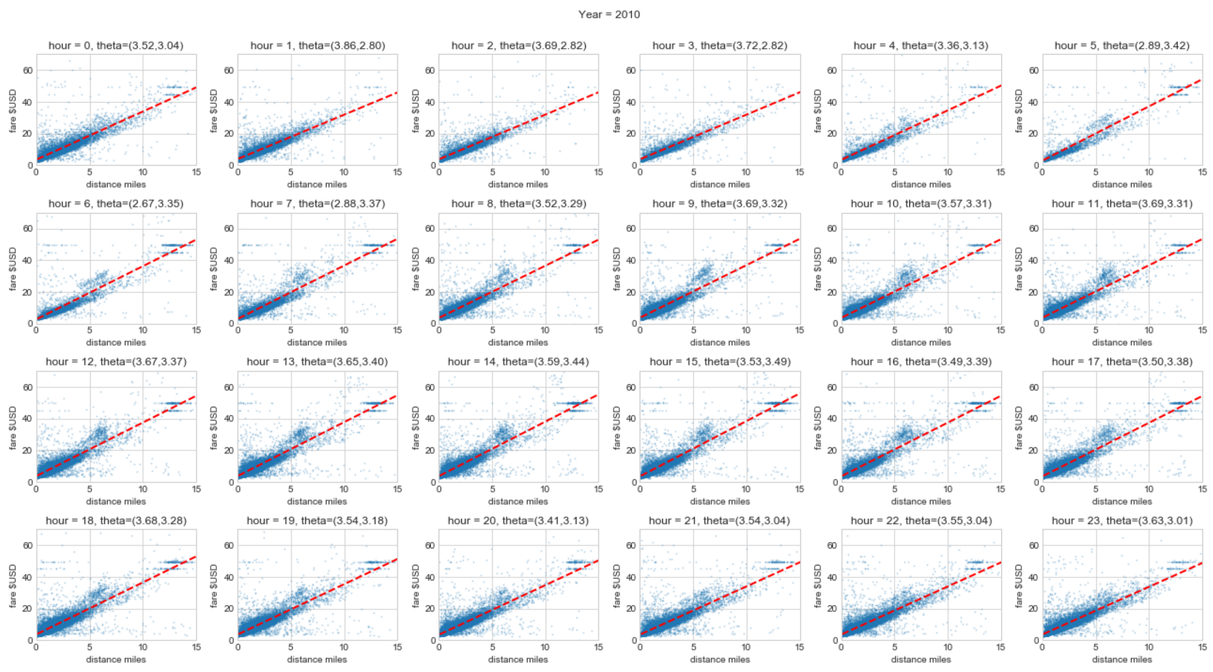


FIGURE 4.6: Linear regression results on cost and distance for 24 hours

We used a single feature (distance) in linear regression as the predictor variable and cost as the response variable. As we observe in figure 4.6, the coefficients for all the regression are positive, which indicates that predictor and response variables are positively correlated. In our regression models, the coefficient for distance varies between 2.67 to 3.86 for different hours. The constant factor also varies between 2.8 to 3.49. As the results of linear regression suggest, for different hours, the correlation between distance and cost varies. Based on our observation in EDA for temporal features, the following features are added to the data points.

- Pickup year
- Pickup month
- Pickup week of the year
- pickup weekday
- Pickup day

- Pickup hour

After expanding the temporal feature from **pickup datetime** to mentioned six features, we consider spatial features and their relation with trip cost and trip duration. The spatial features include **pickup longitude, pickup latitude, dropoff longitude, dropoff latitude**. First, the scatter plot is used to show the pickup locations.

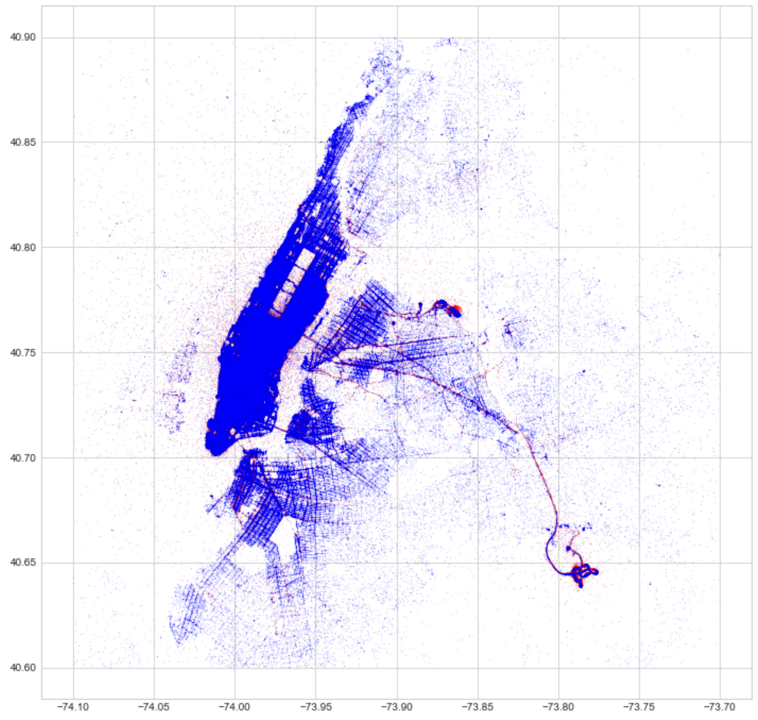


FIGURE 4.7: Scatter plot of pickup locations

Figure 4.7 shows the scatter plot of the pickup location in the dense area after removing far points. Afterward, we plot the distribution of longitude and latitude for both the pickup and dropoff which are visualized in figure 4.8.

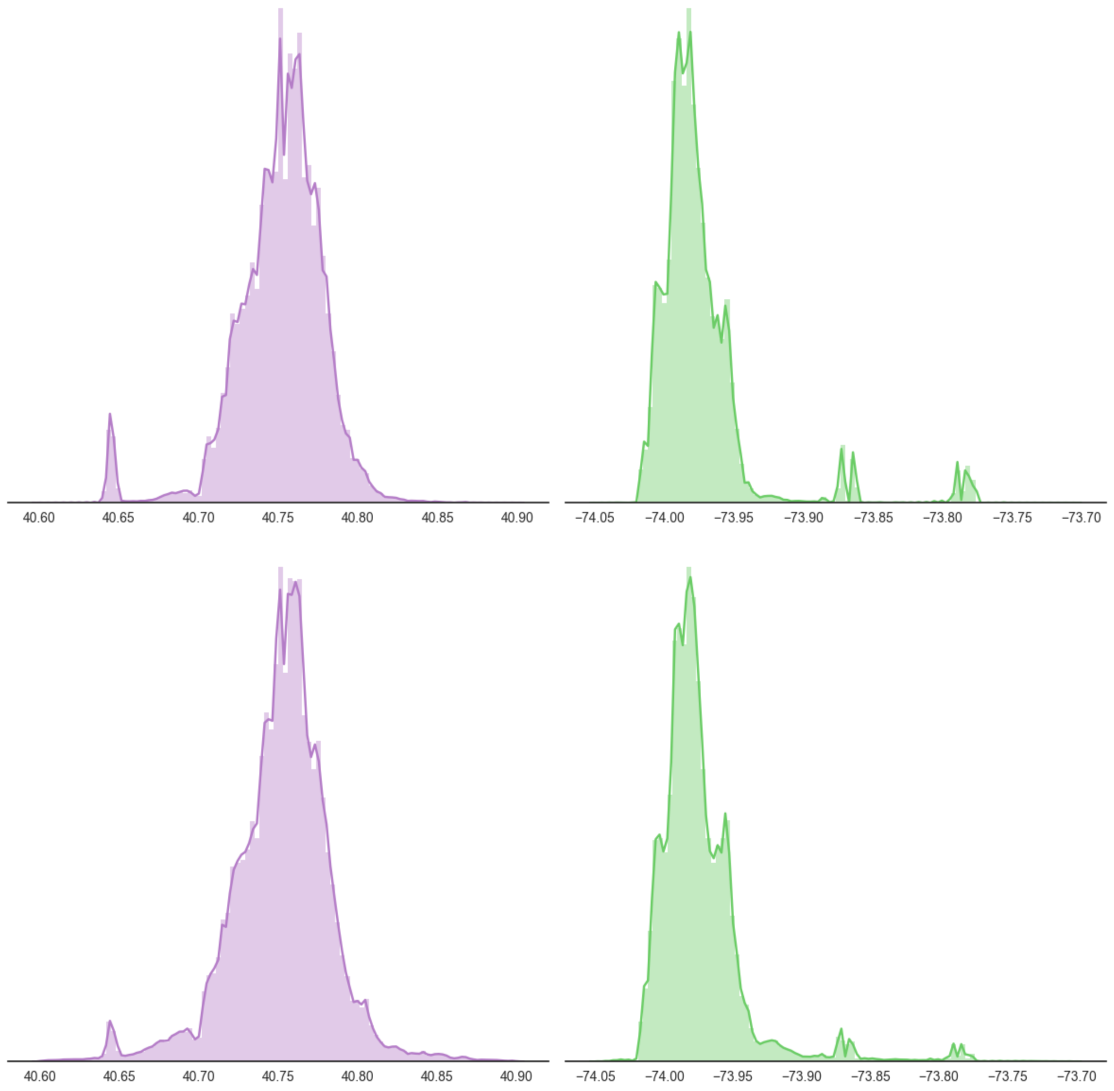


FIGURE 4.8: Latitude and longitude distribution of the pickup and dropoff

We choose three days including Monday, Friday, and Sunday, to plot the density of the requests overtime for 24 hours to observe how the density changed over time from 00 AM to 11 PM.

We found that the pickup location of the requests varied for different days. For example, in Sunday, hot spots indicated by a white spot in 4.11 are different from the hot spots on weekdays like Monday and Friday in figures 4.9 and 4.10. These density plots also provide information about the temporal features, and as we can see, the number of requests on Sunday midnight is much higher than Friday and Monday midnight. It is important to note

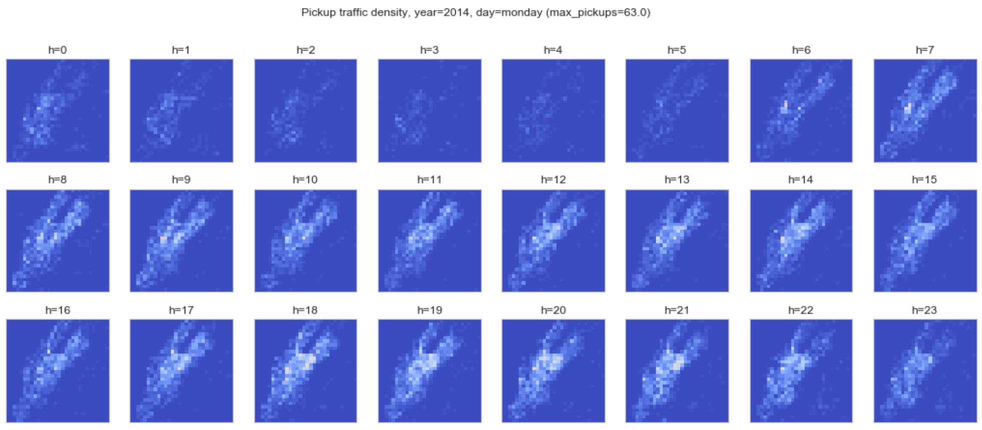


FIGURE 4.9: Pickup density on Monday

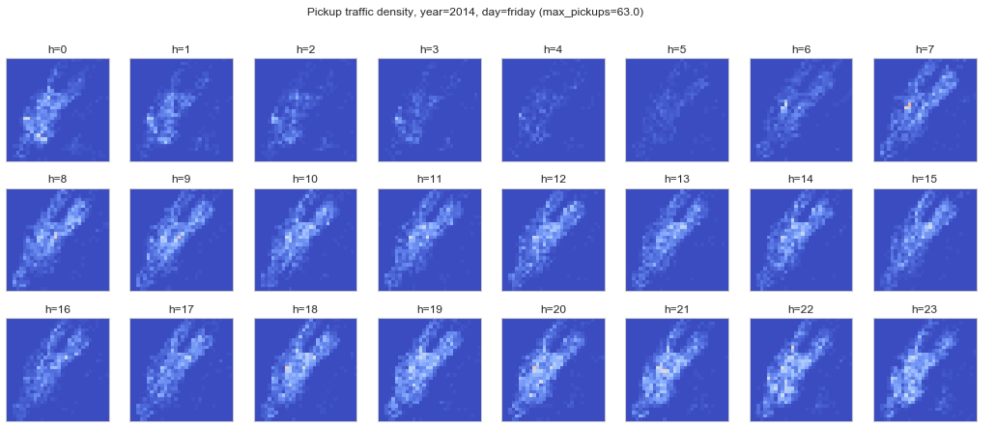


FIGURE 4.10: Pickup density on Friday

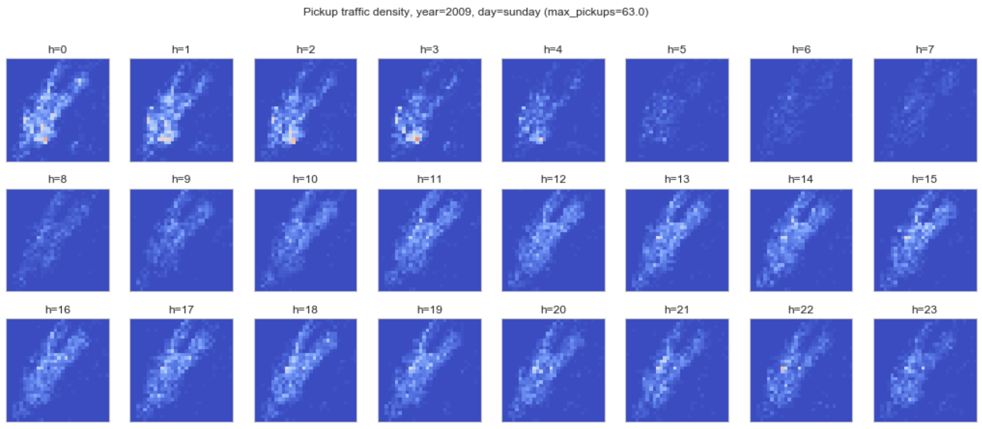


FIGURE 4.11: Pickup density on Sunday

that the density of requests can significantly contribute to finding a match. For example, during peak hours in weekdays in Manhattan, due to a large number of passengers finding a ride with similar spatial and temporal identity is easier than midnight in weekdays where the number of requests is relatively lower.

One of the most important features in predicting trip duration and trip cost is trip distance. Trip distance takes different forms with respect to how we define the distance between two locations. In this work, we used Open Source Routing Machine (OSRM), a routing engine that provides the exact distance between two points based on the Open Street Map (OSM). OSRM receives a trip request with pickup, dropoff, and time. Furthermore, it returns the trip distance for the fastest route and some other features that we explain here and add to features for trip duration prediction.

- Distance: Route distance in meters.
- Motorway, trunk, primary, secondary, tertiary, unclassified, residential: The proportion that spends on the different types of road.
- nTrafficSignals: The number of traffic signals of the road.
- nCrossing: The number of the pedestrian crossing.
- nStop: The number of stop signs.
- nIntersection: The number of the intersection.

In figure 4.12, we plot the OSRM distance on horizontal axis and trip duration on vertical axis. As we can see, there is a positive correlation between distance and duration; in other words, by increasing the OSRM distance, the trip duration increases.

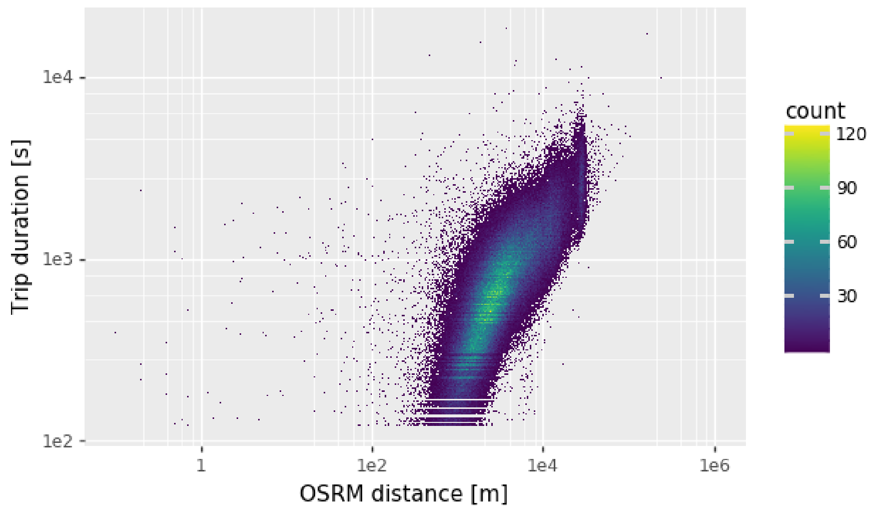


FIGURE 4.12: Scatter plot of OSRM distance and duration

Understanding the relation between spatial features and trip cost is the next step in our analysis. We start by plotting the cost over distance to observe the distribution of data points over the distance-cost plane. In figure 4.13, we plotted the cost over distance for all the data on the left, and on the right, we have the data points with cost under 100$ and distance less than fifteen miles. Observation suggests that there is a strong correlation between distance and cost, but the variation in data is very high, and using other features can help us to handle the variation by representing the data points in a higher dimension space.

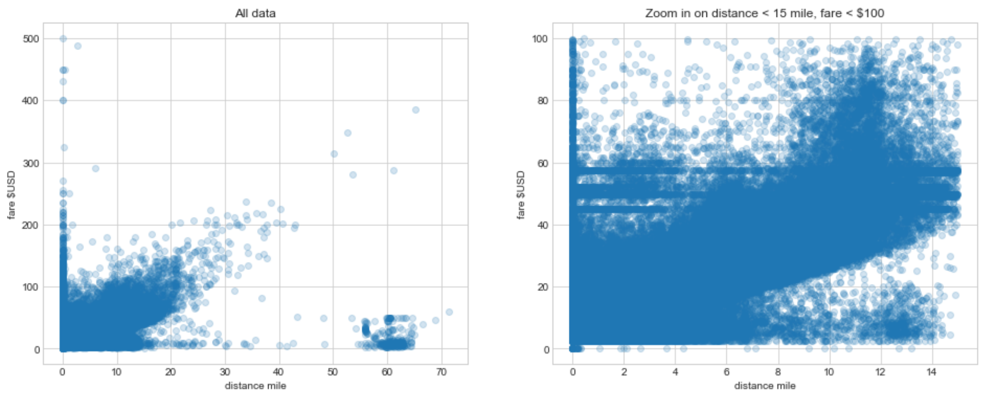


FIGURE 4.13: Scatter plot on cost and distance

Based on the information that we get from figure 4.13, we remove some of the data points as follows. First, the training data consists of elements with zero distance and non-zero cost. It could be related to trips with the same pickup and dropoff. The information about the trip duration and trip distance is not reachable in this case, and we remove them from the dataset. There is a cluster around 50 miles with relatively low costs. These points could be related to discounted trips and are removed from the dataset. Besides the removed points, we also notice that horizontal lines in the right plot could be the fixed cost from a specific location like JFK airport.

To investigate more about the trips from well-known places like airports, we plot the histogram of the data from/to a short distance (1.5 miles) from New York City airports. Figures Figs. 4.14 to 4.16 show that the cost distribution for trips within a 1.5 distance from three NYC airport. The left side includes trips from selected airports and, the right side includes trips to selected airports. The trip cost distributions are much denser than usual places and suggest that for airports and well-known specific places, cost calculated differently.

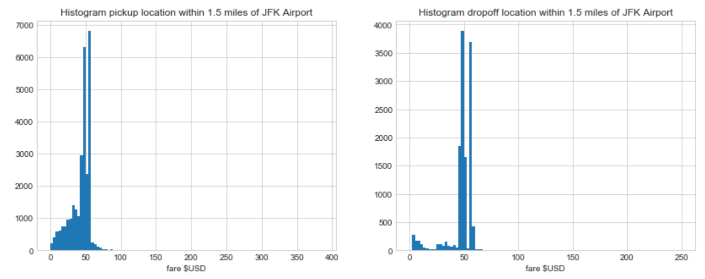


FIGURE 4.14: JFK airport cost histogram

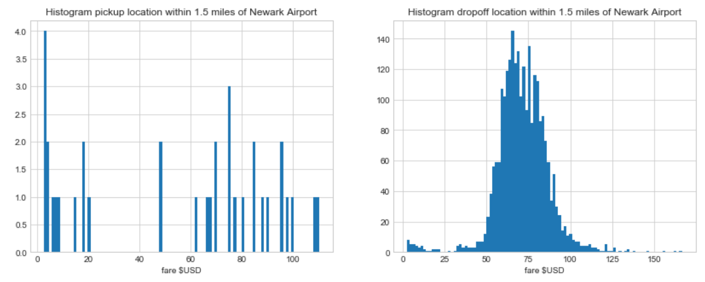


FIGURE 4.15: Newark airport cost histogram

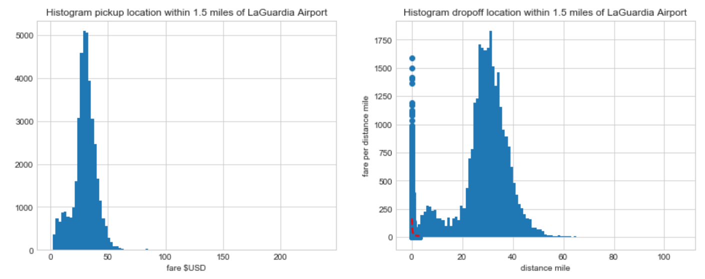


FIGURE 4.16: LaGuardia airport cost histogram

In order to further analyze the data for extracting features, we focus on understanding the effect of direction on the cost. We start by visualizing the cost with respect to differences in pickup latitude and dropoff latitude and pickup longitude and dropoff longitude.

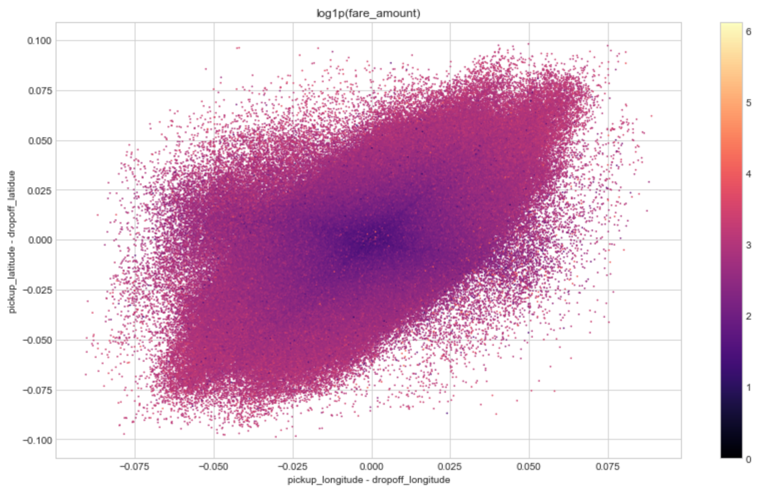


FIGURE 4.17: Correlation between differences in pickup and dropoff and cost (log scale)

From figure 4.17 shows the direction of the trips affect the cost, and it could be varied over cities with different grid forms. We investigate the effect of direction by calculating the precise direction for the trip that is visualized using histogram.

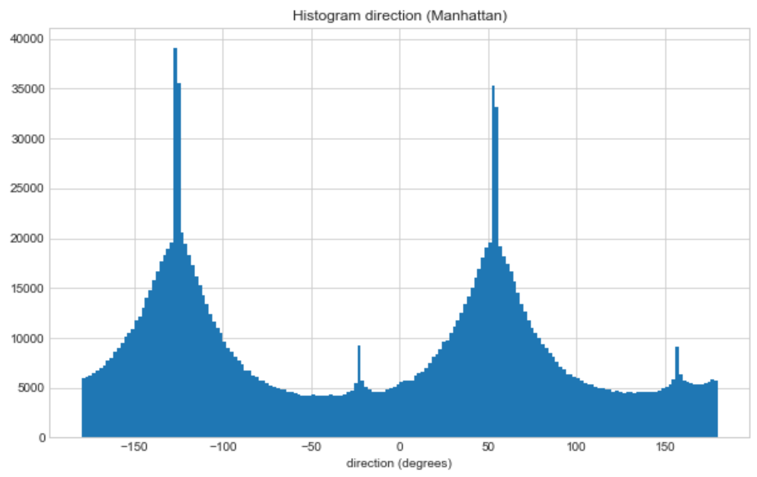


FIGURE 4.18: Histogram of trips direction

Figure 4.18 shows two peaks in the histogram in 60 degrees and -120 degrees. This is the exact angle which Manhattan street has with the horizon. We grouped the trips based on their trip degree and calculate the average trip cost, which is visualized in the following figure. Figure 4.19, shows the average cost for trips. We observed that trips with directions equal to 60 or -120 had the smallest average cost. Based on the dependency between spatial

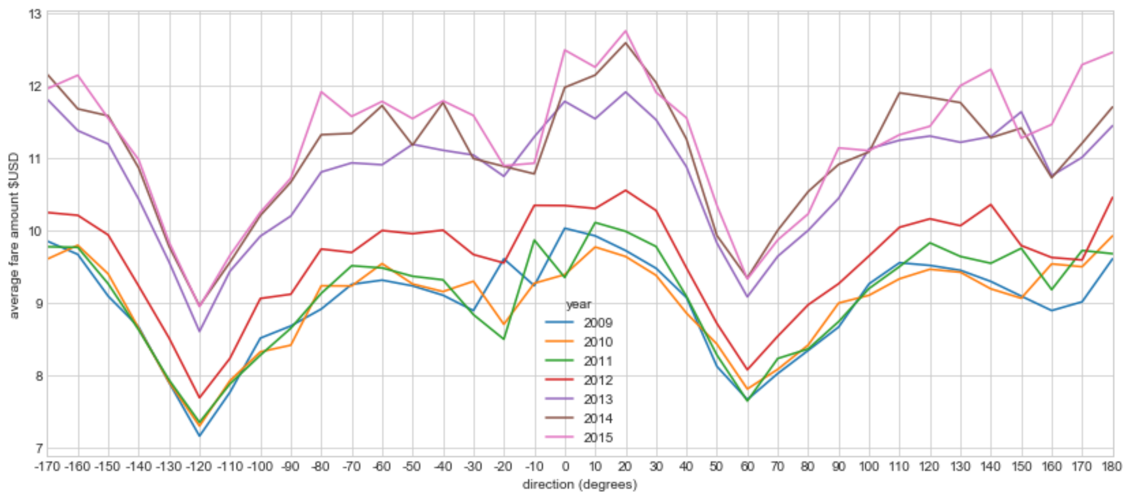


FIGURE 4.19: Average trips cost by degree

features and trip cost, the following features are added to assist the learning procedure for cost prediction.

- **Jfk dist**- Distance to John F. Kennedy airport.
- **Ewr dist**- Distance to Newark Liberty International airport.
- **Lga dist**- Distance to LaGuardia airport.
- **Sol dist**- Distance to Statue of Liberty.
- **nyc dist**- Distance to New York City center.
- **bearing**- Bearing between pickup and dropoff.
- **delta lon**- The difference between pickup and dropoff longitude.
- **delta lat**- The difference between pickup and dropoff latitude.
- **direction**- The angle of moving from pickup point to dropoff point.

In this section, we add 27 features to the data for representing the data points in a higher dimension space in order to train a regression model with acceptable performance and relatively lower error. The added features are categorized into three groups, namely route features, temporal features, and spatial features with 12, 6 and, 9 features for each data point. In the next two sections, we train gradient boosting trees on the data for predicting the trip duration and trip cost.

4.3 Learning Results on Duration and Cost Prediction

In this section, two models are trained separately for duration prediction and cost prediction. After training the regression model for duration prediction, predicted duration is added to data points and used as a feature for predicting the trip cost. Dataset is thus split into three disjoint sets, namely training set, validation set and, test set with the proportions equal to 70%, 10% and, 20 %, respectively. After splitting the data, the training set includes 5.6 million instances; the validation set includes 0.8 million instances, and the test set includes 1.6 million instances. For the evaluation metric, **Root Mean Square Error** is used which can be expressed as the following formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

RMSE is used as the objective function for both regression task (trip duration and trip cost), and the learning task goal is to minimize **RMSE** by training a set of trees for prediction. R^2-score is used to indicate the performance of the regression model to explain the variation in target values (trip duration and trip cost) and is calculated as follows:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where SS_{tot} is the total sum of squared and it is calculated as $SS_{tot} = \sum_i (y_i - \bar{y})^2$ and $SS_{res} = \sum_i (y_i - \hat{y}_i)^2$, which $\hat{y}_i$ is the predicted value for sample i. The R^2-score can be varied between the 0 and 1 and higher value means a better fit to the data.

As discussed earlier, in gradient boosting trees, a set of hyperparameters must be set before starting the learning procedure. These hyperparameters include learning rate, tree depth, feature fraction, bagging fraction, the frequency for bagging, λ_{L_1} and, λ_{L_2}. Bayesian optimization explores the hyperparameters space on their defined bound to find the parameters that maximize the objective function. In the regression task, the training goal is to minimize the loss error and to convert it to a maximization problem, negative of the loss is maximized in hyperparameters tuning using Bayesian optimization.

Bayesian optimization considers 32 different settings for hyperparameters, and the one with minimum loss is the optimal hyperparameters for training the gradient boosting trees. As mentioned earlier, gradient boosting trees use a series of weak learners (single tree) to construct a powerful model. For both trip duration and trip cost, 10000 trees are trained to minimize the regression loss. After learning the trees, we describe the crucial features in data that gradient boosting trees used to split. Next, we provide the learning curve and evaluation results (RMSE and R^2) on the test set for duration and cost prediction models.

4.3.1 Results of trip duration prediction

Before starting the learning procedure, optimal hyperparameters are selected by using Bayesian optimization. The optimal hyperparameters set for training the gradient boosting trees is the one with minimum loss in Bayesian optimization in table 4.1. The learning rate is 0.1, and the feature fraction is 0.8202, λ_{L_1}, and λ_{L_2} are equal to 3.754 and 1.866 respectively, bagging frequency, and subsample fractions are 2 and 0.99, the last parameter is the maximum depth of the tree which is 16.

The model is trained with the optimal hyperparameters and plots the loss during the training process on both the validation set and training set. In figure 4.20, *RMSE* is calculated for the training set and validation set over learning iteration (trees). The error converges to a relatively low number (below 5) at the end of learning, and before mentioning the results on the test set, features importance is discussed on trees formation.

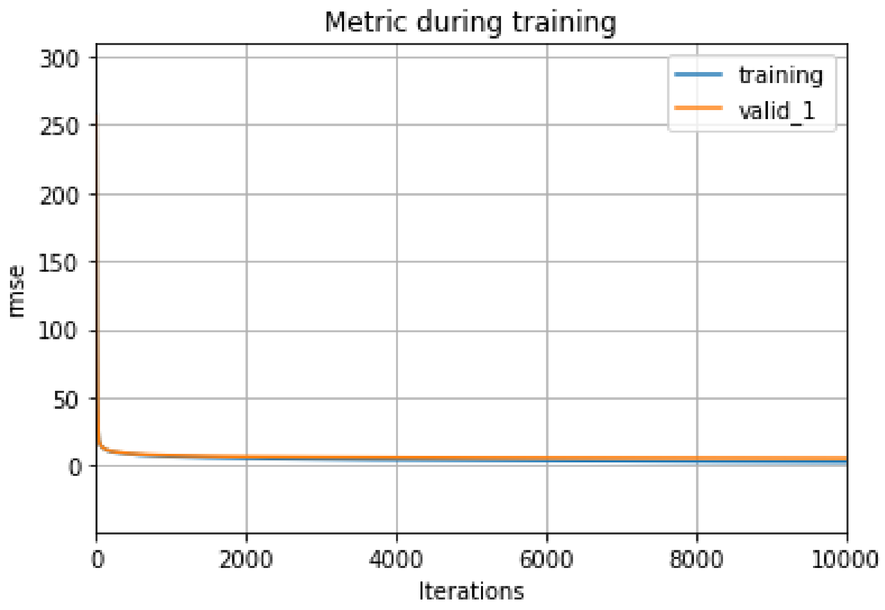


FIGURE 4.20: Train set and validation set loss for duration prediction

As mentioned earlier, regression trees use features to split the data into subgroups. In the duration prediction task, feature importance is calculated, and the top fifteen features are as follows:

iter.	loss	feature frac.	λ_{L_1}	λ_{L_2}	LR	depth	subsample	bagging freq
1	5.269	0.5205	2.049	2.944	0.4525	13.48	0.5851	9.462
2	4.195	0.7977	1.678	2.208	0.2605	13.35	0.6371	6.94
3	4.117	0.5402	0.1933	3.376	0.114	16.76	0.9356	4.225
4	4.09	0.7839	4.396	2.262	0.3599	16.18	0.9719	3.734
5	4.074	0.9548	2.5	0.194	0.1519	13.47	0.642	5.414
6	4.398	0.8786	3.534	2.64	0.4383	13.09	0.6541	2.515
7	3.946	0.713	3.884	2.672	0.1635	14.89	0.8366	3.298
8	3.843	0.8202	3.754	1.866	0.1003	16.71	0.9993	1.947
9	4.528	0.9569	1.553	2.381	0.4673	16.49	0.5275	9.698
10	4.141	0.9777	0.3474	2.403	0.2673	14.47	0.7172	3.499
11	4.114	0.6576	3.07	1.994	0.1664	15.7	0.7688	9.55
12	4.264	0.7344	1.269	0.9706	0.454	14.4	0.8383	5.414
13	3.949	0.853	1.536	0.4828	0.1487	14.28	0.8536	3.389
14	4.293	0.6826	1.33	2.889	0.2358	16.0	0.5411	2.542
15	4.526	0.5689	2.251	2.164	0.3726	16.22	0.9529	9.468
16	4.2	0.5407	2.788	4.56	0.03645	15.85	0.7179	4.002
17	4.055	1.0	5.0	0.0	0.025	17.0	0.5	10.0
18	4.075	1.0	5.0	5.0	0.025	17.0	0.5	10.0
19	3.862	1.0	0.0	0.0	0.025	17.0	1.0	1.0
20	4.447	0.5	5.0	0.0	0.025	14.99	0.5	1.0
21	3.866	1.0	5.0	5.0	0.025	17.0	1.0	1.0
22	4.037	1.0	5.0	0.0	0.025	13.0	0.5	10.0
23	3.866	1.0	5.0	5.0	0.025	13.0	1.0	6.21
24	4.306	0.5	0.0	0.0	0.025	13.0	1.0	1.0
25	4.025	1.0	5.0	2.074	0.025	14.69	0.5	7.444
26	3.861	1.0	0.0	5.0	0.025	17.0	1.0	1.0
27	3.86	1.0	0.0	5.0	0.025	13.0	1.0	5.008
28	3.86	1.0	0.0	5.0	0.025	13.0	1.0	1.0
29	4.335	0.5	5.0	5.0	0.025	13.0	1.0	1.0
30	3.972	1.0	0.0	0.0	0.025	17.0	0.5	4.546
31	4.019	1.0	0.0	0.0	0.025	17.0	0.5	10.0
32	3.982	1.0	0.0	0.0	0.025	13.0	0.5	7.59

TABLE 4.1: Bayesian optimization on trip duration prediction

1. Route features:

    - Distance is the most important feature for splitting.
    - Tertiary is the 4th important feature for splitting.
    - Secondary is the 5th important feature for splitting.
    - Trunk is the 6th important feature for splitting.
    - nTrafficSignals is the 11th important feature for splitting.
    - Residential is the 15th important feature for splitting.

2. Spatial features:

    - Pickup latitude is the 7th important feature for splitting.
    - Dropoff latitude is the 8th important feature for splitting.
    - Pickup longitude is the 9th important feature for splitting.
    - Dropoff longitude is the 10th important feature for splitting.
    - Direction is the 14th important feature for splitting.
    - Jfk dist is the 13th important feature for splitting.

3. Temporal features:

    - Pickup day is the 2nd important feature for splitting.
    - Week of the year is the 3rd important feature for splitting.
    - Pickup hour is the 12th important feature for splitting.

Based on the features' importance, route features contribute the most in trees splitting and dividing data into smaller subgroups for the regression task. The route characteristics, including distance, road type, and traffic signals, help the model to reduce the loss. From temporal features, pickup day and week of the year are among the top three features. Among the spatial features, pickup and dropoff information (longitude and latitude) are among the top ten features for splitting.

After training the model on the training set, the model's performance is evaluated on unseen data (test set) to report the capability of the model for prediction on the unseen data points. The prediction results for the test set are available in the table 4.2.

R^2	RMSE
0.9995	5.8036

TABLE 4.2: Evaluation results on test set

The R^2-score is very close to one and shows the model's ability to explain the variance in the test set. The RMSE loss for the test set is low, and the model can accurately predict the trip duration. The huge number of trees in gradient boosting trees helps the model to be able to explain the high variance in target value. The trained model is regarded as a function that takes a trip, including the pickup and dropoff information (longitude and latitude) and the request time. As output, it returns the trip duration for a specified trip. It is used in our ridesharing model to calculate the trip duration for passengers, denoted by $\hat{\phi}(.)$ in simulation.

4.3.2 Results of trip cost prediction

Here we aim to train a model for cost prediction. As mentioned earlier, we calculated the trip duration by the function $\hat{\phi}(.)$, added to features and used in trip cost prediction. The size of the training set, validation set, and test set is the same as trip duration prediction. First, we define a boundary for each of the hyperparameters and Bayesian optimization search to find the optimal hyperparameters from the bounded search space. The learning rate boundaries are 0.025 and 0.5, the feature fraction and the bagging fraction are bounded between $[0,1]$. The regularization coefficients, λ_{L_1} and λ_{L_2}, can take values between 1 and 5. The depth of the tree takes a value between 13 to 17, and the bagging frequency could vary between 1 to 10. In the following table, results for Bayesian optimization are provided, and the one with minimum loss is selected as the optimal hyperparameters for training the model.

In table 4.3, Bayesian optimization is used to find the optimal hyperparameters for the model. Iteration number 17 has the minimum loss and is selected as the optimal hyperparameters. The gradient boosting is trained on the training set, and here we can see the validation set and train set loss during the training procedure.

iter	loss	feature fraction	λ_{L_1}	λ_{L_2}	LR	depth	bagging frequency
1	4.139	0.5205	2.049	2.944	0.4525	13.48	2.535
2	4.066	0.9701	2.997	1.678	0.2348	14.98	1.791
3	3.986	0.6371	3.3	0.4016	0.04336	15.7	2.686
4	4.096	0.9697	4.356	1.791	0.2947	16.52	5.072
5	4.039	0.8525	3.981	4.719	0.1693	16.64	5.5
6	4.028	0.5194	1.336	0.582	0.1599	14.96	7.815
7	3.995	0.8534	2.64	4.351	0.03592	14.23	2.515
8	4.036	0.713	3.884	2.672	0.1635	14.89	7.059
9	4.04	0.6277	3.202	3.754	0.2022	13.63	9.341
10	4.047	0.993	0.5259	4.569	0.1726	14.9	9.381
11	4.158	0.9364	0.2751	4.832	0.4788	13.28	5.325
12	4.026	0.755	1.843	2.172	0.1569	14.26	6.525
13	4.087	0.6994	1.488	3.371	0.2804	16.8	5.218
14	4.026	0.6269	0.9706	4.516	0.1912	15.71	5.414
15	4.034	0.853	1.536	0.4828	0.1487	14.28	7.365
16	4.087	0.6327	0.6327	1.33	0.2994	14.77	7.747
17	3.966	0.5	5.0	5.0	0.025	13.0	1.0
18	3.973	0.5	5.0	5.0	0.025	17.0	1.0
19	3.975	0.5	0.0	0.0	0.025	17.0	1.0
20	3.985	0.5	5.0	5.0	0.025	13.0	5.69
21	3.978	0.5	0.0	0.441	0.025	13.0	10.0
22	3.98	0.5	5.0	5.0	0.025	17.0	10.0
23	3.976	0.5	5.0	0.0	0.025	13.0	1.0
24	3.976	0.5	5.0	0.0	0.025	13.0	10.0
25	3.968	0.5	0.0	5.0	0.025	17.0	1.0
26	3.977	0.5	0.0	0.0	0.025	17.0	10.0
27	3.972	0.5	0.0	0.0	0.025	13.0	1.0
28	3.973	0.5	5.0	0.0	0.025	17.0	10.0
29	3.975	0.5	5.0	0.0	0.025	17.0	1.0
30	3.975	0.5	5.0	5.0	0.025	13.17	10.0
31	3.977	0.5	2.175	0.0	0.025	14.11	5.125
32	3.971	0.5	0.0	5.0	0.025	13.0	1.0

TABLE 4.3: Bayesian optimization on trip cost prediction

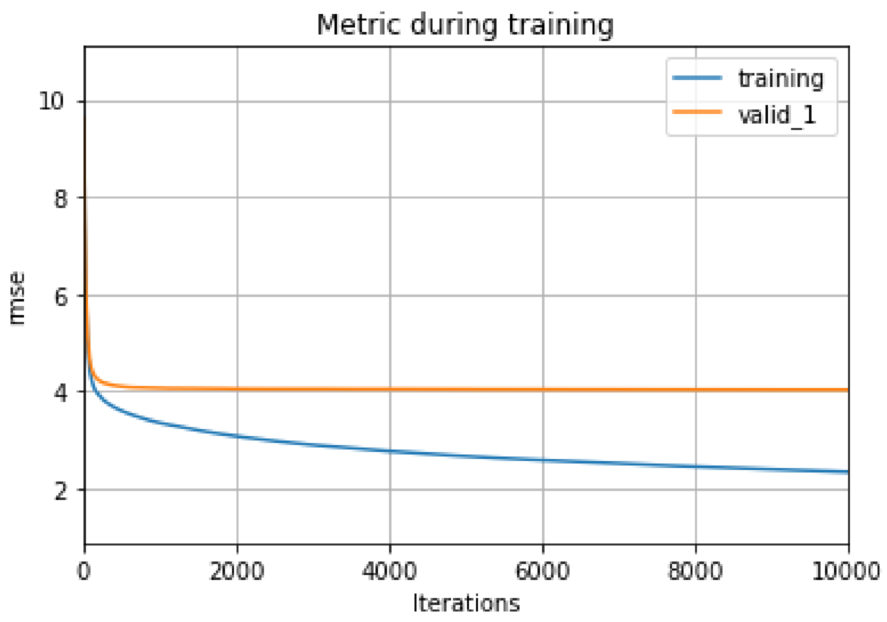


FIGURE 4.21: Train set and validation set loss for cost prediction

One of the main advantages of a tree-based learning algorithm for both continuous (regression) and discrete (classification) prediction is that they are hard to over-fit. In our training process, train loss and validation loss decrease, and the model is not over-fitted to the training set. After training the gradient boosting trees for the cost prediction task, features importance is discussed to indicate the top fifteen features that contribute most in trees forming.

1. Route features:
   - Distance is the most important feature for splitting.
   - Duration is the 2nd important feature for splitting.
2. Spatial features:
   - Pickup latitude is the 3rd important feature for splitting.
   - Dropoff latitude is the 5th important feature for splitting.
   - Pickup longitude is the 6th important feature for splitting.
   - Dropoff longitude is the 7th important feature for splitting.
   - Jfk dist is the 4th important feature for splitting.

- Direction is the 14th important feature for splitting.
- Delta longitude is the 9th important feature for splitting.
- Delta latitude is the 10th important features for splitting.
- Lga dist is the 11th important features for splitting.
- Sol dist is the 13th important features for splitting.

3. Temporal features:

   - Pickup hour is the 8th important feature for splitting.
   - Pickup day is the 12th important feature for splitting.
   - Pickup year is the 15th important feature for splitting.

The two most important features are the trip distance and trip duration. It is important to note that the route characteristics, including tertiary, secondary, trunk, nTrafficSignals, and residential, are encoded in trip duration, and the model eliminates them in splitting. The model is evaluated on the test set, and the results are available is table 4.4.

R^2	RMSE
0.8548	3.6954

TABLE 4.4: Prediction result for trip cost

The RMSE is relatively low, which shows that the trained trees can predict the trip cost accurately. The other evaluation metric R^2 shows the trees' ability to explain the variance of the target feature (trip cost) in the test set. The trained model used to predict the trip cost is denoted as $\hat{C}(.)$, which takes a trip and returns the trip cost.

Two important functions in ridesharing simulation are approximated in this chapter. In the next chapter, we define the simulation environment, and both cost function and duration function are used to indicate the trip cost and trip duration on the edges of the city network. Dijkstra's shortest path algorithm is used to calculate the shortest travel time for a trip. These two functions help us to capture the dynamics in the city network with respect to cost and duration. Both functions play an important role in providing an accurate simulation for ridesharing.

Chapter 5

Ridesharing Simulation

During this chapter, our ridesharing approach is simulated using NYC yellow-cab data. To run the simulation, we first provide the algorithm for our ridesharing procedure based on chapter 3 and afterward prepare the data for simulation. As [90] mentioned, the static case results for ridesharing is an upper bound for results in the dynamic case. Based on this observation, instead of forecasting future demand, we assume that a perfect model is available and forecast the exact demand for the near future (within 10 minutes from processing time). In the remaining of this chapter, we provide the exact algorithm for our ridesharing approach in section 5.1, and afterward, we provide the simulation environment details from both spatial and temporal perspective in section 5.2 and finally in section 5.3, we present the simulation results for our ridesharing approach. In section 5.3, the different aspects of adaptive waiting time and its influence on the matching rate in the simulation are investigated.

5.1 Ridesharing Algorithm

In this section, we provide our ridesharing algorithm for a defined time horizon T on a one-minute interval. In our ridesharing algorithm, the inputs are ε, $G_{m,n}$, $\hat{C}(.)$, and $\hat{\phi}(.)$ which are detour flexibility, city network, cost function and duration function. The algorithm output is a set of matched passengers who share their ride during the time horizon. In this work, to mimic dynamic ridesharing's online nature, data is streamed in a discretized manner (one-minute interval), and future demands are just used to calculate waiting time for passengers.

```
Algorithm 3: Ridesharing algorithm
input  : - The city network $G_{m,n}$ with size $m \times n$
         - A detour flexibility factor ($\varepsilon$)
         - $\hat{C}(.)$ as the cost function
         - $\hat{\phi}(.)$ as the duration function
output: A matching set $M$ that is iteratively updated
Data: Request set $D$, which is streamed over the time horizon
1 for $t \in \{0,\dots,T\}$ do
2     Calculate the trip duration between adjacent vertices in $G_{m,n}$ based on their
        centers' longitude and latitude at time $t$.
3     Add the unmatched passengers from $\Pi_{t-1}$ to $\Pi_t$, $\Pi_t \leftarrow \Pi_{t-1}$.
4     calculate trip duration for $P = \{r_i \in D | t_i = t\}$ by Dijkstra's algorithm.
5     Calculate the optimal waiting time for $P = \{r_i \in D | t_i = t\}$ based on equation 5.
6     Indicate the leaving candidate as follows: $L_t = \{r_i \in \Pi_t | t_i + w_i = t\}$.
7     Create the passengers' graph $G_p$ and add edges for feasible matches.
8     Construct matching $M_t$ using greedy approach in algorithm 1 and remove
        matched passengers from $\Pi_t$.
9     $M \leftarrow M \cup M_t$
```

At each iteration, the ridesharing algorithm updates the weights on the city network based on the trip duration, from the vertex's center to an adjacent vertex's center. Afterward, the shortest path (in terms of duration) is calculated using Dijkstra's algorithm and thus added to features for each request. By knowing the shortest trip path, the upper bound for waiting time is calculated, and subsequently, the optimal waiting time is calculated and added to features by equation 5. The leaving candidate set is defined afterward as the set of passengers in the passengers' pool, whose waiting time has elapsed and are now ready to match. After creating the passengers graph, the greedy algorithm 1 finds a matching set M on it. Adjacent vertices in M accomplish a ride together.

5.2 Simulation Environment

We explained the ridesharing algorithm in the last section, and in this section, we describe the simulation environment precisely. By the simulation environment, we mean all the factors that shape the ridesharing, including the detour flexibility factor, the city graph $G_{m,n}$, D, which represents requests, and time horizon T. First, we define the requests set D. We consider the NYC yellow-cab data for our simulation [93]. We have selected our data from January 2012 for the simulation. The dataset consists of more than 14 million ride requests.

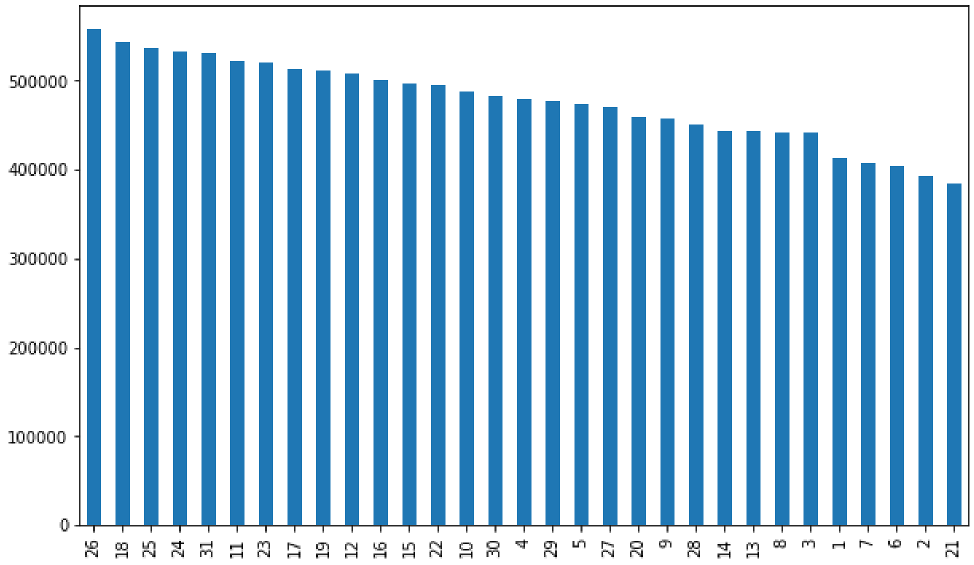


FIGURE 5.1: Number of requests per day in January 2012

In figure 5.1, January 26th is observed as the busiest day with 557203 trip requests. We select the January 26th data for the ridesharing simulation, and we eliminate the other days and explore it more. To observe the hourly change in the number of requests, the number of requests over time is plotted and shown in figure 5.2.

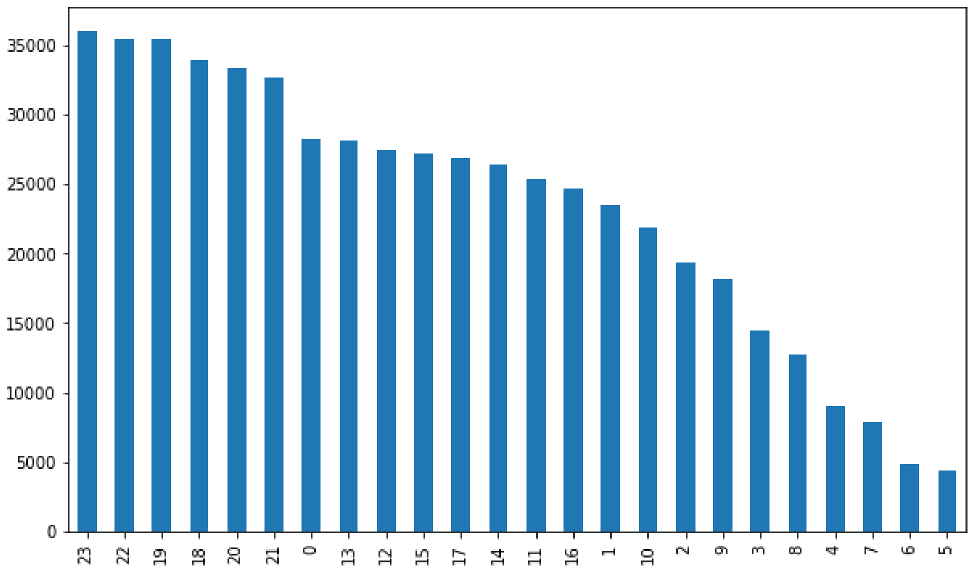


FIGURE 5.2: Number of requests over hours in January 26th

The number of requests is significantly varying over the hours. On January 26th, which is a Sunday, the rush hours were in the late hours between 18 to 23, with more than 30000 requests per hour, and the lowest number of requests belonged to Sunday early morning between 5 and 6 with less than 5000 requests. This variation in the number of requests would reveal the importance of critical mass in ridesharing. In our experiments, ridesharing works for $T = 1440$, which covers a whole day in one-minute intervals. After defining the temporal aspects of the simulation environment, we present the spatial aspects of the ridesharing simulation. The spatial aspects of ridesharing define the geographical area where ridesharing can occur.

To define the spatial aspect, first, we plot the distribution of longitude and latitude for both pickup and dropoff, and afterward, we define a rectangle on New York City, where the ridesharing can occur. In figure 5.3, we observe the distribution of longitude (green distribution) and latitude (purple distribution) for pickup (top) and dropoff (bottom).

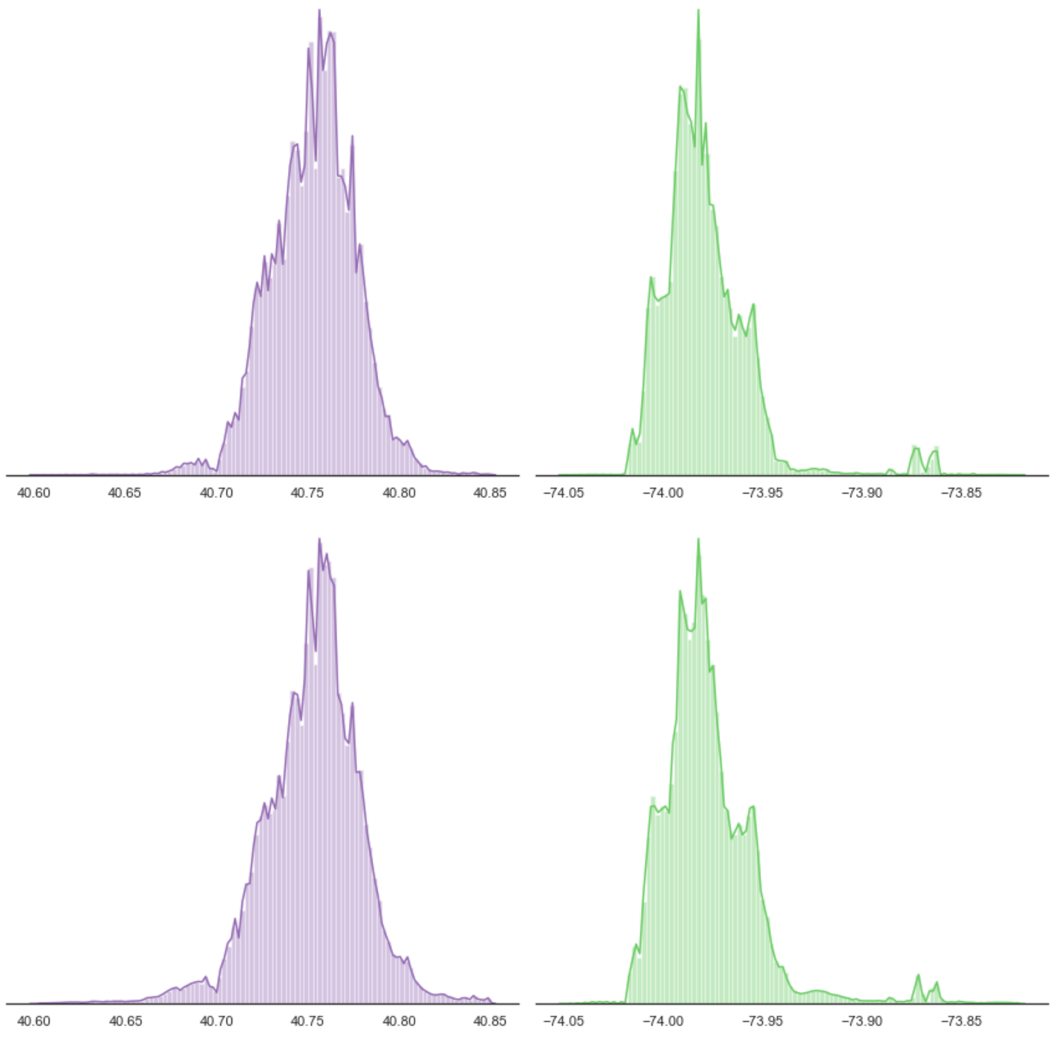


FIGURE 5.3: Longitude and latitude distribution for pickup and dropoff

We define a rectangle with an area of 44 km^2. The square is defined on latitude between 40.7 and 40.8 (length equal to 11 km) and longitude between -74.1 and -73.96 (length of 4 km). We define the dataset D as the set of the inner rectangle trips which includes 356049 ride requests. The figure 5.4, shows the number of inner rectangle requests over hour. We observe the same pattern in figure 5.2.

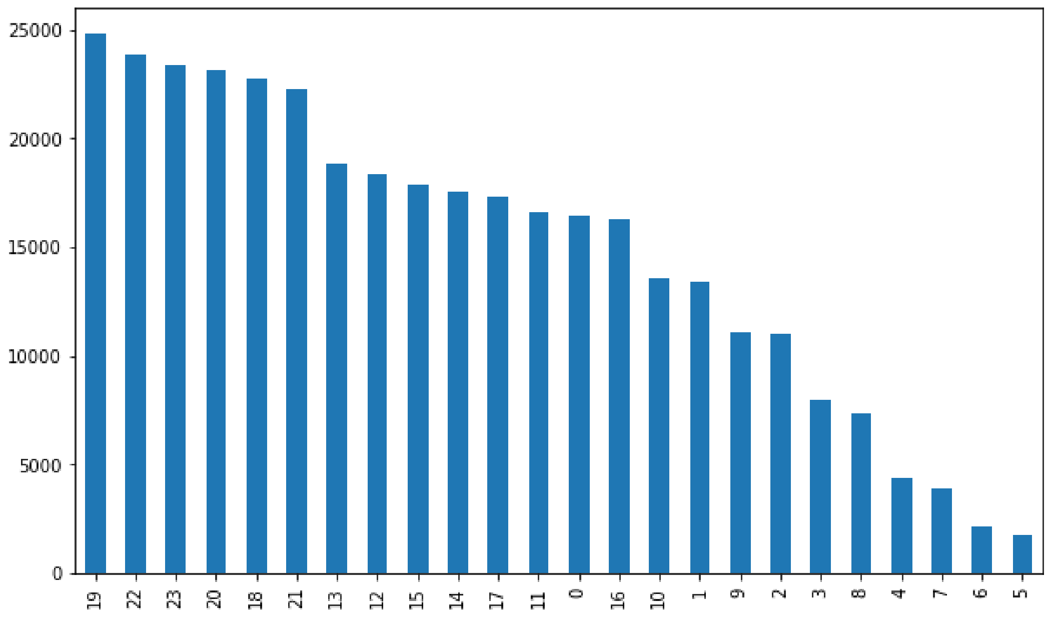
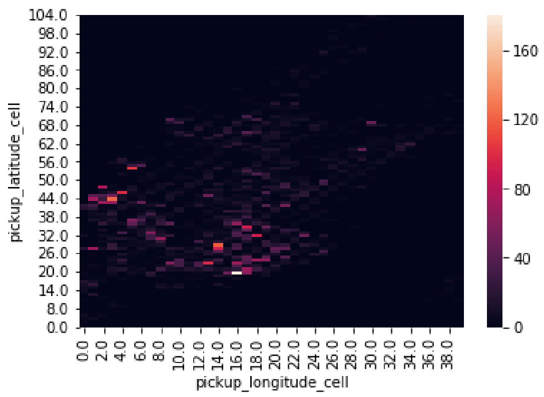
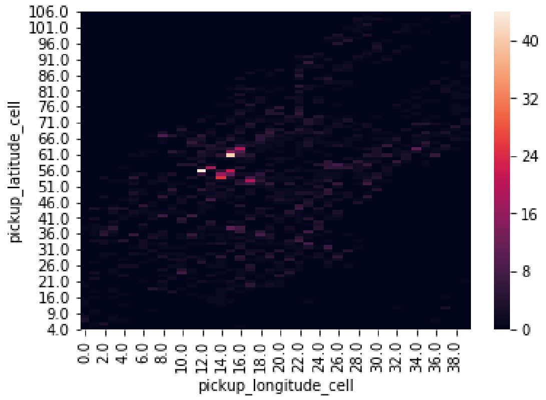


FIGURE 5.4: Number of requests in ridesharing rectangle

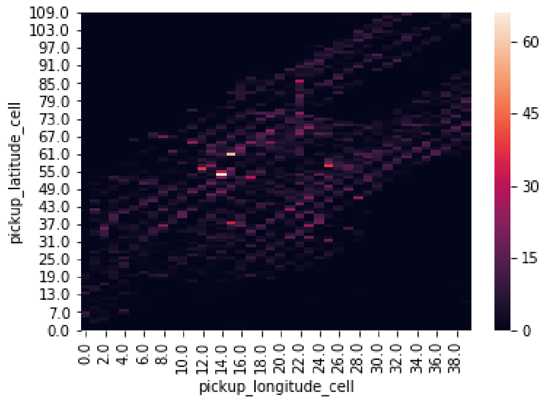
We define the city network $G_{m,n}$ on the rectangle by partitioning the rectangle into $100m \times 100m$ square using 110×40 square tile. In other words, the city network is a rectangle partitioned to square and denoted by $G_{110,40}$. We assume that the trips in each square are identical in spatial terms and come from the center of the square. The distance between two neighbor squares (those sharing a side) is 100 meters and at each time step $t \in \{0,\ldots,T\}$, function $\hat{\phi}(.)$ is used to predict the trip duration between the neighbor pairs of squares and the shortest path, in terms of travel duration calculated on $G_{110,40}$ using Dijkstra's algorithm, where weights represent the trip duration between squares. The distance and duration of a path are added to features of a request, and the trip cost is then calculated based on these features and approximated using the function $\hat{C}(.)$.



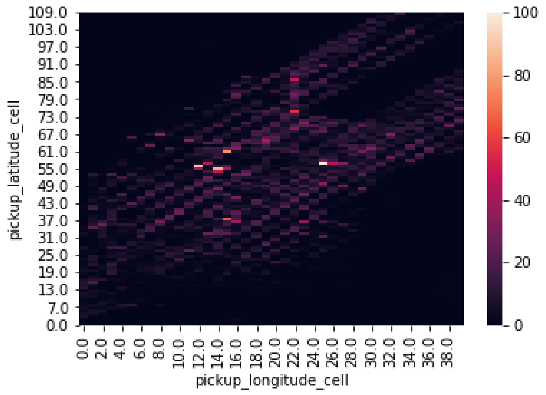
(A) The cumulative number of requests at 2



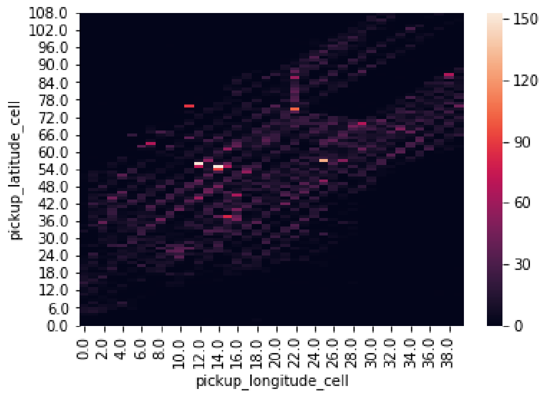
(B) The cumulative number of requests at 6



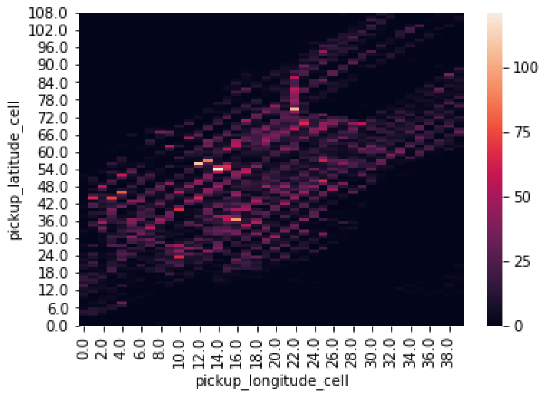
(C) The cumulative number of requests at 8



(D) The cumulative number of requests at 10



(E) The cumulative number of requests at 13



(F) The cumulative number of requests at 19

FIGURE 5.5: Heat map of requests on city network $G_{110,40}$ for six hours

In figure 5.5, the city network is visualized by plotting the cumulative number of requests for six hours. As described earlier, we suppose the upcoming requests available to calculate the optimal waiting time. For two vertices $v, w \in G_{110,40}$ at time t, we can calculate the probability of a request which comes from v and goes to w at time t ($p_{v,w,t}$) by dividing the number of request from v to w at time t by the total number of requests at time t and subsequently calculate the optimal waiting time by minimizing the expected travel cost using equation 5. It is important to measure and analyze the quality of the solution provided by our ridesharing system. We used well-known metrics for ridesharing based on the metrics provided by [3, 90]. Here we define the metrics that are used to analyze the quality of the ridesharing system. To measure the performance of our ridesharing framework, we have measured the following metrics:

1. Matched percentage
2. Reduction in total traveled distance
3. Reduction in the cumulative trip cost
4. Mean of travel overhead
5. Mean of pickup time
6. Average waiting time

The percentage of the matched passengers measures the ability of the ridesharing framework. One of the environmental advantages of ridesharing is reducing the total traveled distance by vehicles that lead to less greenhouse gas emissions. To observe our ridesharing framework's performance in reducing the total traveled distance, we calculate the total traveled distance with and without ridesharing as follows. In the case in which passengers travel alone, vehicles traveled distance is denoted by $L_{v_{solo}} = \sum_{r_i \in D} l(\phi_i)$ and in the case which sharing is allowable, the vehicles traveled distance was calculated as the sum of the overall trips belongs to M and is denoted by $L_{v_{shared}}$. The reduction in vehicles traveled distance is also calculated as $1 - \frac{L_{v_{shared}}}{L_{v_{solo}}}$.

From the passengers' perspective, ridesharing can reduce the traveling cost. Cumulative cost is calculated with and without ridesharing to observe our model's performance in cost reduction. The cumulative cost is the cost that all the passengers in D paid to travel from their origins to destinations. In the case that passengers traveled alone, the cumulative cost denoted by $C_{solo} = \sum_{r_i \in D} C(\phi_i, t_i)$ and in the case that passengers can share their rides, C_{shared} is equal to the sum of the cost for overall path in sharing. The reduction in cumulative trip cost is calculated as $1 - \frac{C_{shared}}{C_{solo}}$. The average travel duration in the case which passengers travel solo is denoted by $Dur_{solo} = \frac{\sum_{r_i \in D} |\phi_i|}{|D|}$ and in the case that sharing is allowable it is equal to the average of travel time from sending request to reach the destination, and it is denoted by Dur_{shared}. The mean of travel overhead is calculated as $Dur_{shared} - Dur_{solo}$. The average waiting time is equal to the average of assigned waiting time and is denoted by $\bar{w} = \frac{\sum_{r_i \in D} w_i}{|D|}$. The mean of pickup time is equal to mean of the time that passengers wait before picked up by a vehicle. These three metrics measure the deviation of passengers from their shortest path.

In this thesis, the simulation runs for different values of ε from 0.3 to 0.8 to observe the detour flexibility factor's effects. Detour flexibility factor plays a crucial role in ridesharing, large epsilon leads to larger detour from the shortest path which can be divided to waiting time and actual trip path and in this way increase the chance of finding a ride, but at the same time, it reduces the passenger convenience by increasing the overall trip duration. On the other hand, for small detour flexibility factor, deviation from the shortest path is small and subsequently leads to short scheduling flexibility, and it is hard to find a match due to limitation in time.

5.3 Simulation Results

In this section, we provide the ridesharing simulation results and waiting time function behavior on the defined simulation dataset explained. After analyzing the assigned waiting time from temporal and spatial aspects, matching results are provided to observe our ridesharing system's ability to share a ride between two passengers. Before mentioning the results, it is essential to note that our ridesharing procedure is done in an online manner on January 26th from 00 AM to 11:59 PM; On the defined rectangle with an area of 44 km^2 and for $\varepsilon \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$.

5.3.1 Waiting Time Function Behavior

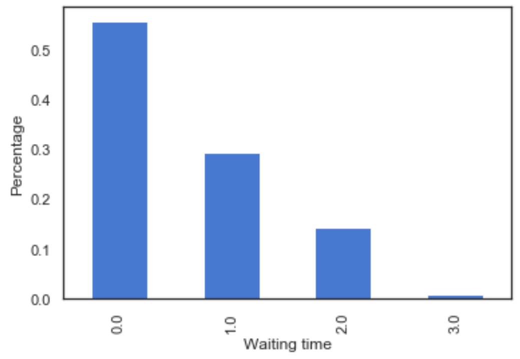
The novelty of this work is defining an adaptive waiting time function that has used the future demands in the near future to minimize the expected cost for the passengers. It also encourages the riders to participate in the ridesharing system and achieve critical mass. In this section, the waiting time calculated using algorithm 3. After calculating the waiting time for all the requests during the data stream, we discuss the spatial and the temporal aspects of waiting time.

The analysis of the waiting time begins by providing the mean and the standard deviation of waiting time for different values of ε in table 5.1. Based on the values of the mean and the standard deviation for ε between 0.3 to 0.8, the mean and the standard deviation increase by increase in ε.

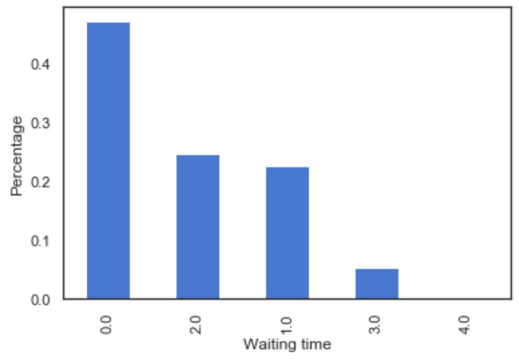
ε	mean	std
0.3	0.59646	0.75011
0.4	0.8843	0.9654
0.5	1.18871	1.18596
0.6	1.55855	1.44817
0.7	1.93007	1.69645
0.8	2.23448	1.88855

TABLE 5.1: The mean and the std of waiting time based on ε

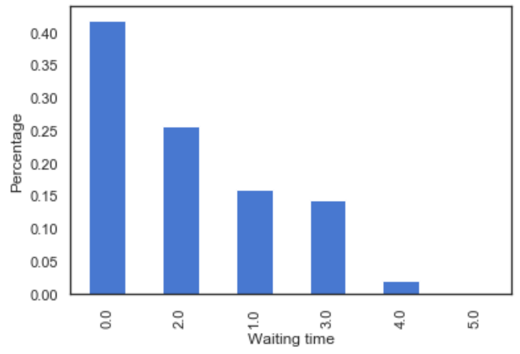
To analysis more of the relation between ε and waiting time, the bar charts of the waiting time values and their percentage are plotted for different values of ε and shown in Figs. 5.6 and 5.7. The figures show that by increasing ε, the waiting time boundaries expanded. For example, for $\varepsilon = 0.3$, the waiting time is between 0 to 3 minutes, and for $\varepsilon = 0.8$, the waiting time is between 0 to 8 minutes.



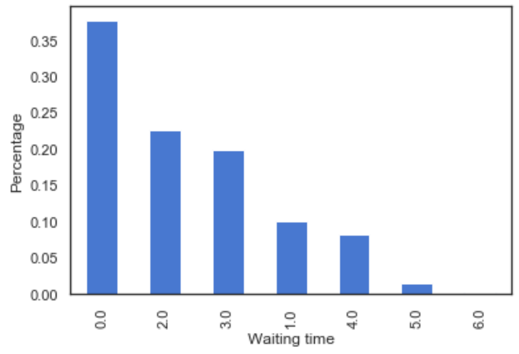
(A) Waiting time bar chart for $\varepsilon = 0.3$



(B) Waiting time bar chart for $\varepsilon = 0.4$

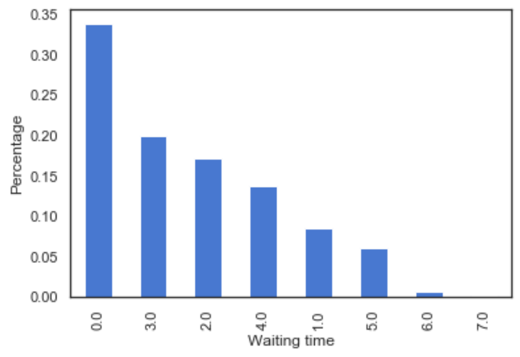


(C) Waiting time bar chart for $\varepsilon = 0.5$

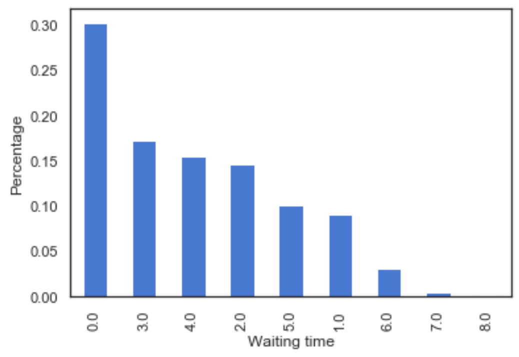


(D) Waiting time bar chart for $\varepsilon = 0.6$

FIGURE 5.6: Waiting time bar chart for ε between 0.3 to 0.6



(A) Waiting time bar chart for $\varepsilon = 0.7$



(B) Waiting time bar chart for $\varepsilon = 0.8$

FIGURE 5.7: Waiting time bar chart for ε between 0.7 to 0.8

The bar charts in Figs. 5.6 and 5.7 are briefly discussed here. For $\varepsilon = 0.3$ in figure 5.6a, the waiting time range varies between 0 to 3, and waiting time for more than 50 percent of passengers is equal to zero. As ε increases, the number of passengers with waiting time equal to zero decreases, and in figure 5.7b, around thirty percent of the passengers have a waiting time equal to zero. To better understand the temporal behavior of waiting time function, the error bar, which includes mean and std of the waiting time, is plotted hourly for ε between 0.3 to 0.8, and the observed results are discussed here.

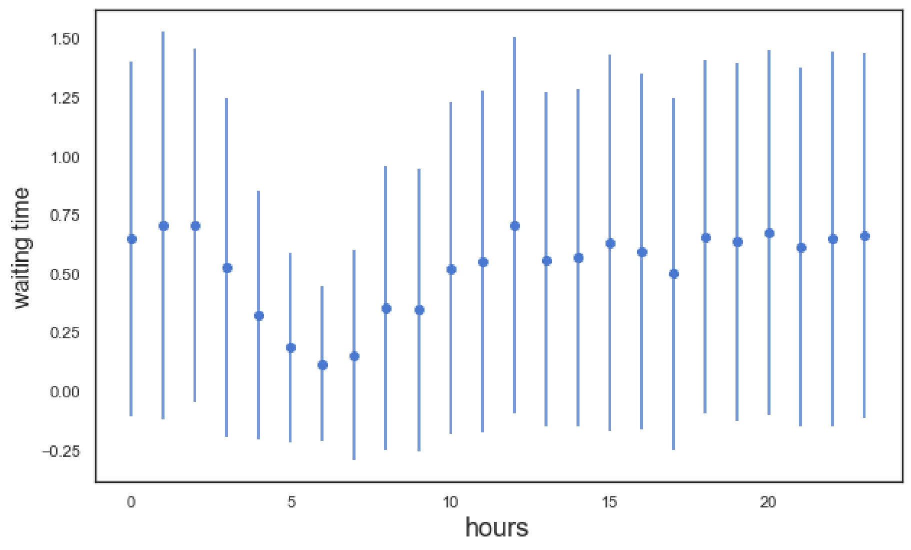


FIGURE 5.8: Hourly waiting time mean and std for $\varepsilon = 0.3$

In figure 5.8, the mean and standard deviation of waiting time are plotted hourly, and here we mention our observation. First, the mean of waiting time shows a pattern related to the number of requests. In other words, for hours with a higher number of requests, the waiting time is relatively higher. It is since waiting in rush hour can lead to finding a good match as a significant number of requests comes available over time. We observe this pattern for rush hours between 0 to 2 and 18 to 23. For hours with a small number of requests, the waiting time is relatively shorter, for the waiting in unbusy hours cannot lead to finding a match in a short period.

Next, the error bars are plotted for ε between 0.4 to 0.8, and we discussed the waiting time behavior over the hour.

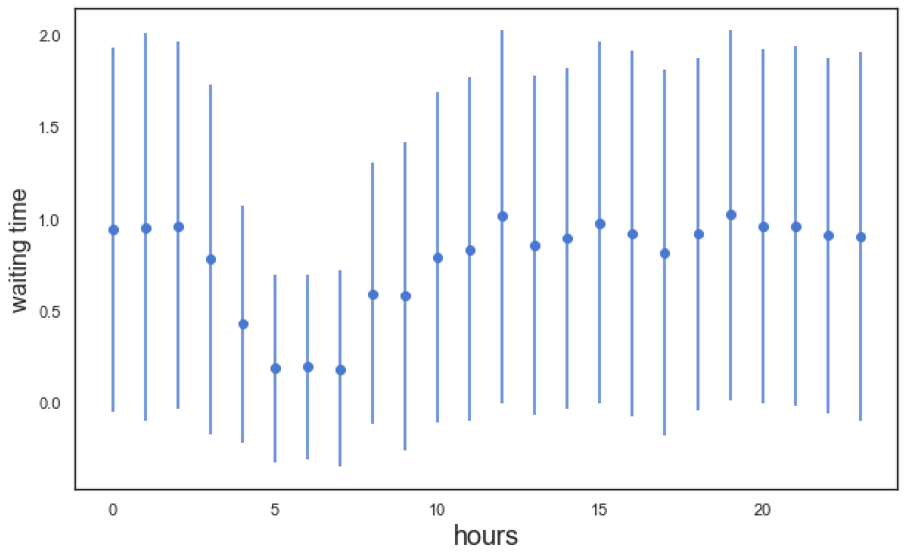


FIGURE 5.9: Hourly waiting time mean and std for $\varepsilon = 0.4$

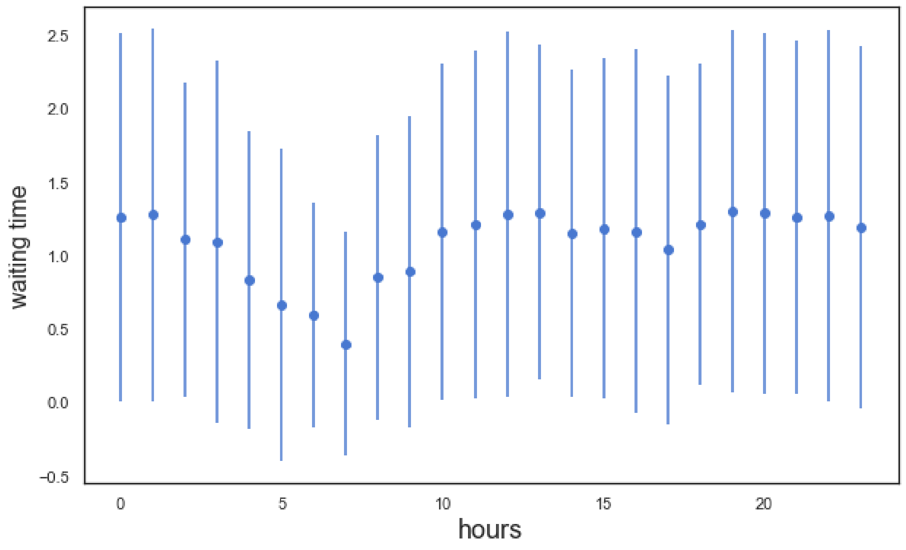


FIGURE 5.10: Hourly waiting time mean and std for $\varepsilon = 0.5$

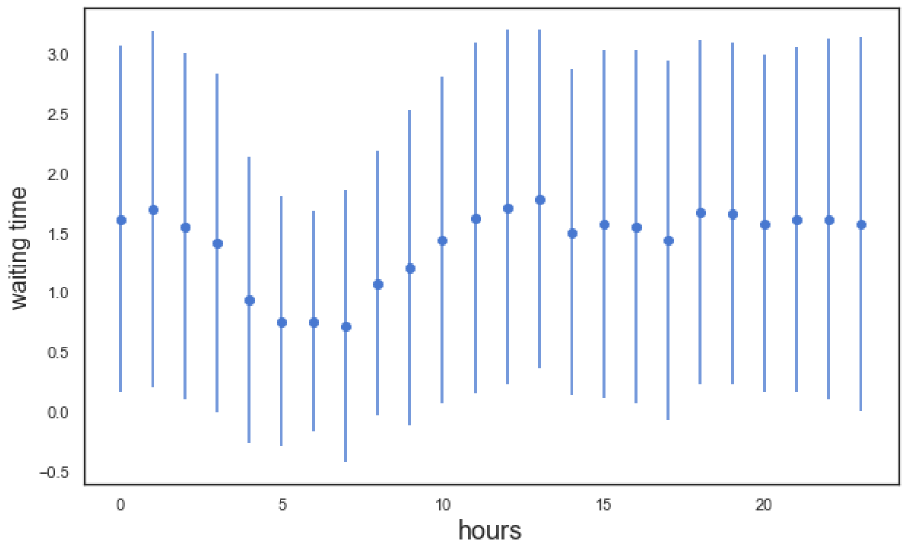


FIGURE 5.11: Hourly waiting time mean and std for $\varepsilon = 0.6$

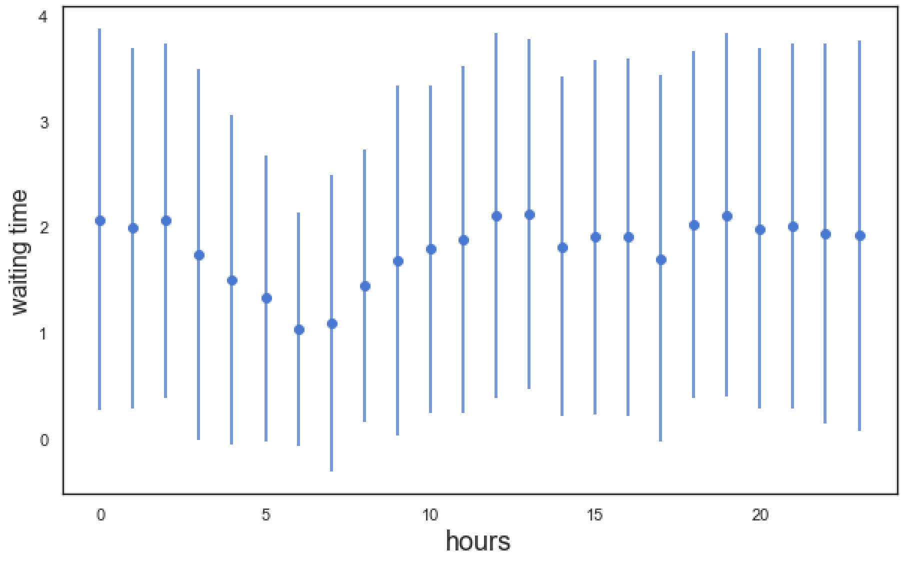


FIGURE 5.12: Hourly waiting time mean and std for $\varepsilon = 0.7$

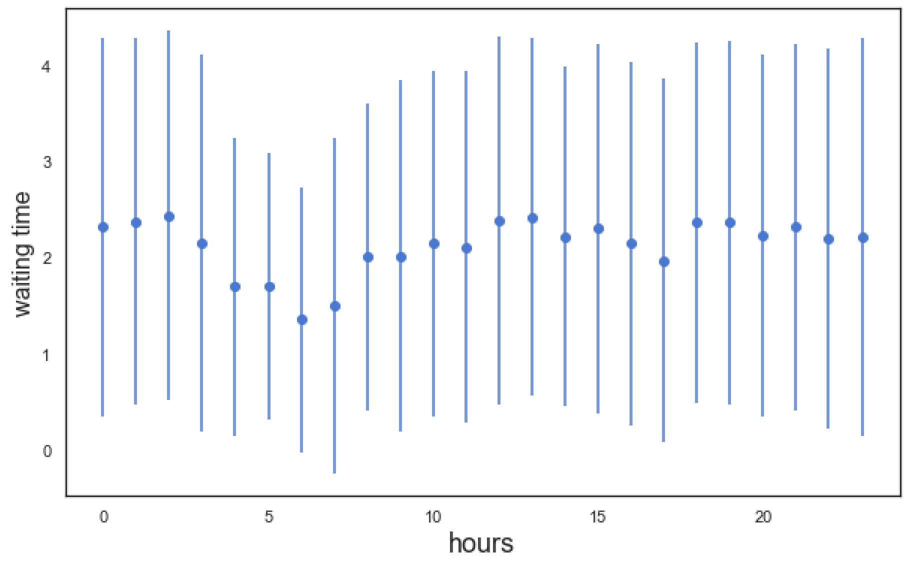
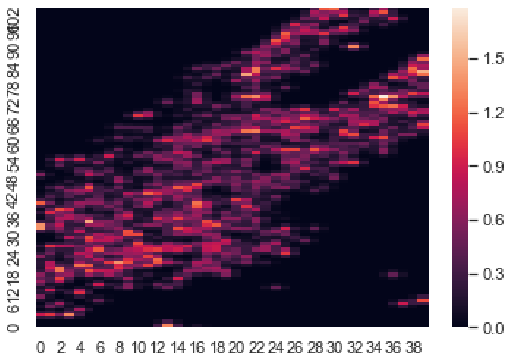
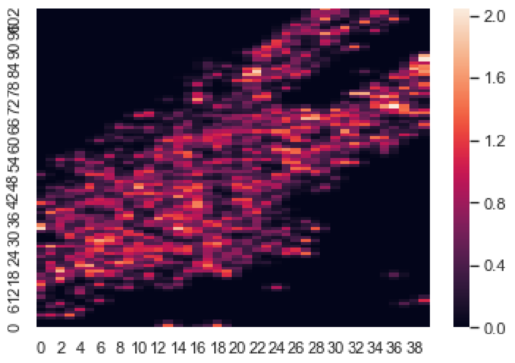
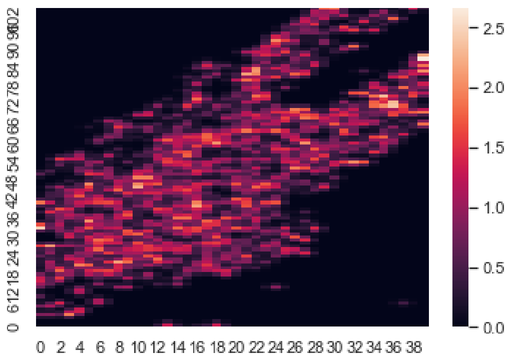


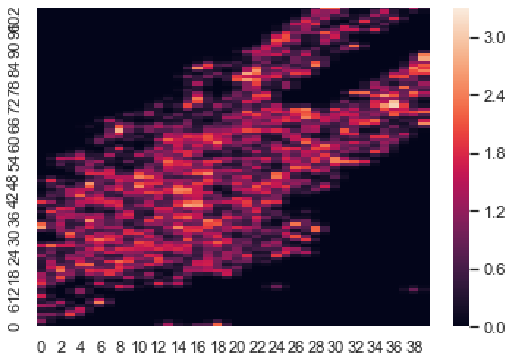
FIGURE 5.13: Hourly waiting time mean and std for $\varepsilon = 0.8$

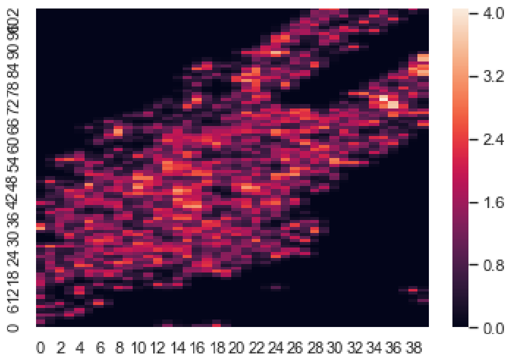
In Figs. 5.9 to 5.13, We observe that by increasing ε from 0.4 to 0.8, the waiting time increased in average and standard deviation. For hours with a relatively smaller number of requests, we observe that increasing the ε leads to moving from a V-shaped behavior to a U-shaped behavior in those hours. It is important to note that the short waiting time is a double-edged sword that can occur for two reasons. First, the current passengers pool is rich enough to give us a good match and minimize the expected cost. Second, the passengers pool is poor and will remain poor, resulted by the small number of requests that appear in the system. In matching results, we look into the behavior of waiting time and its consequences on the matching rate, which is to be discussed shortly, but before that, the waiting time values are plotted on the city network $G_{110,40}$ to observe the spatial distribution of waiting time.


(A) Waiting time heatmap for $\varepsilon = 0.3$


(B) Waiting time heatmap for $\varepsilon = 0.4$


(C) Waiting time heatmap for $\varepsilon = 0.5$


(D) Waiting time heatmap for $\varepsilon = 0.6$


(E) Waiting time heatmap for $\varepsilon = 0.7$

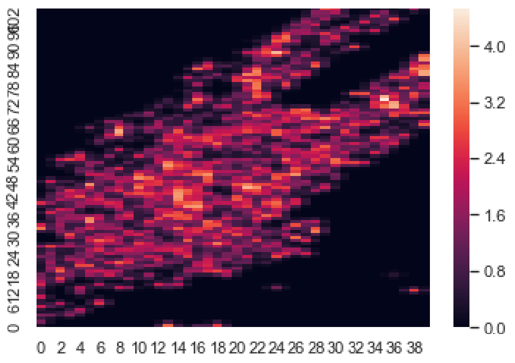

(F) Waiting time heatmap for $\varepsilon = 0.8$

FIGURE 5.14: Mean of the waiting time on $G_{110,40}$ for ε between 0.3 to 0.8

In figure 5.5, We observe a tendency towards zero waiting time in the border of the city and also on the border of Central Park. It is important to note that the mean of waiting time for grids is smooth in a small neighborhood, which shows the spatial dependency between neighborhoods. In figure 5.5, we calculate the mean of waiting time, based on waiting times related to each cell in the city network $G_{110,40}$ and subsequently, the average over the hours which as mentioned earlier reveals different patterns due to the different request flows. It is also important that a better representation of spatial dependency should consider both the pickup and dropoff information and in figure 5.5, only the pickup information is considered.

5.3.2 Matching Results

Here we provide the matching results using greedy algorithm 1 to demonstrate the ability of our ridesharing approach to help passengers to accomplish a trip together. In this work, the matching rate is defined as the percentage of passengers who matched to share their rides. The ability of our ridesharing procedure is measured as follows; first, we calculate the matching rate for different values of ε between 0.3 to 0.8, and second, the matching rate is calculated over hours to be able to understand the effect of the number of requests in matching rate.

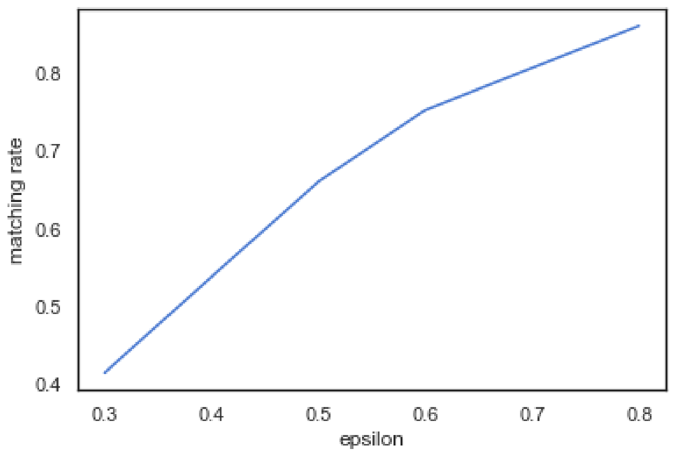


FIGURE 5.15: Overall matching rate for ε between 0.3 to 0.8

In figure 5.15, the overall matching rate over the time horizon is shown for different values of ε. The matching rate for $\varepsilon = 0.3$ is 0.415 and shows that even for small scheduling flexibility in rich (in terms of the number of requests) and bounded area, our ridesharing approach can match more than 40 percent of the passengers. The matching rates for ε between 0.4 to 0.8 are 0.538, 0.66, 0.752, 0.806 and, 0.86, respectively. As we can observe, the matching rate slope decreased over time, which shows the system saturated. This phenomenon shows that in ridesharing, it is hard to match some of the passengers even by increasing the trip duration by 80%. By matching 86% of the passengers using $\varepsilon = 0.8$, less than 50000 of the passengers remained unmatched and needed to travel alone, and more than 306202 of the passengers could share their ride, and it has led to enormous environmental impact which we discuss later.

The hourly matching rate for $\varepsilon = 0.3$ shown in figure 5.16, and it shows a strong dependence between the number of requests and matching rate. The matching rates for 5 and 6 in the morning are 11% and 9%, respectively. By increasing the number of requests, the matching rate increased, and the maximum matching rate belongs to 11 in the morning, January 26th, with around 50% of passengers matched.

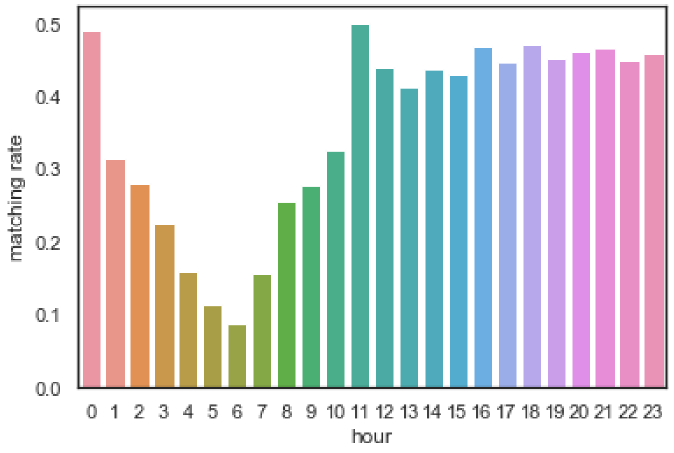


FIGURE 5.16: Hourly matching rate for $\varepsilon = 0.3$

In figure 5.17, the matching rate for different values of waiting time is shown for $\varepsilon = 0.3$. The matching rate for waiting time equal to zero is low and less than 20%. On the other hand, the matching rate for waiting time equal to one, two, and three are higher and more than 60%. Figure 5.17 illustrates the ability of the waiting time function to assign waiting time equal to zero in the case that finding a match is hard based on the upcoming request, spatial and temporal aspects of the concerned request and also ε.

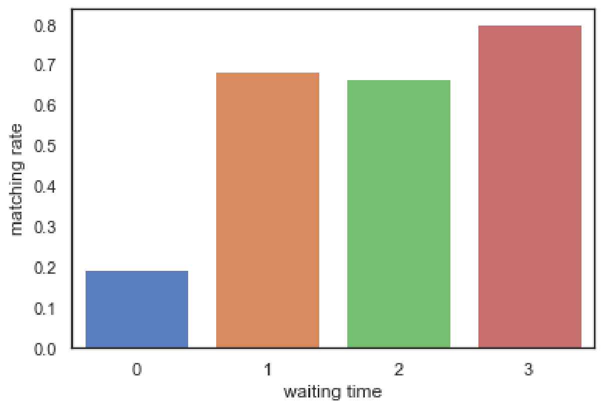


FIGURE 5.17: Matching rate on waiting time for $\varepsilon = 0.3$

The same approach is used to visualize the hourly matching rate and matching rate based on waiting time for ε between 0.4 and 0.8. In figure 5.18, Hourly matching rate for $\varepsilon = 0.4$ is shown. The overall matching rate for $\varepsilon = 0.4$ is 53.8%, and the matching rate is increased by 12.3% compared to $\varepsilon = 0.3$, and the effects are visible in the hourly matching rate. By increasing the ε from 0.3 to 0.4, the matching rate is doubled at the unbusy hours (5 and 6) and reaches to 22%. It is also important to note that the matching rate for busy hours (more than 13000 requests) is more than 50%, and the maximum matching rate belongs to 10 in the morning, with 65% of passengers matched and accomplish a ride together.

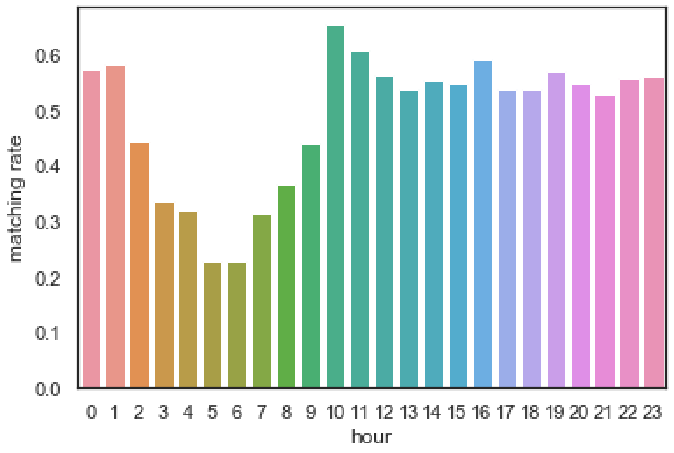


FIGURE 5.18: Hourly matching rate for $\varepsilon = 0.4$

The matching rate versus waiting time is plotted and shown in figure 5.19. By increasing ε from 0.3 to 0.4, the matching rate for passengers with waiting time equal to 0 increased from 19% to 22%. Furthermore, 89% of the passengers with waiting time equal to 4 got matched. The matching rate for waiting time equal to 1, 2, and 3 is 78, 82, and 86, respectively.

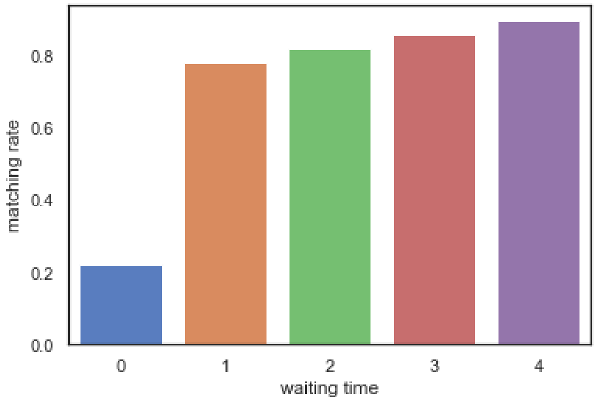


FIGURE 5.19: Matching rate on waiting time for $\varepsilon = 0.4$

Next, the hourly matching rate is plotted in figure 5.20, in which the overall matching rate is 66%. The matching rates for 5 and 6 in the morning, which has the lowest number of requests are 23% and 22%, respectively. The matching rate for these hours increased 2% and presented that the matching is saturated and that it is hard to find a match even by increasing the scheduling flexibility (ε). As the matching rate for busy hours (more than 13000 requests) passed 60%, it showed a significant improvement compared to $\varepsilon = 0.4$. The increase in overall matching rate and also hourly matching rate indicate the importance of scheduling flexibility (ε) in matching and scheduling process.

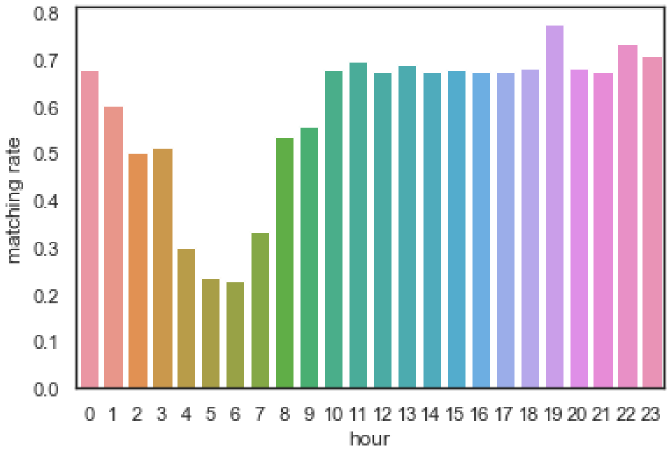


FIGURE 5.20: Hourly matching rate for $\varepsilon = 0.5$

In figure 5.21, the matching rate is shown for different values of waiting time. The matching rate for waiting time between 0 and 5 is 0.33, 0.92, 0.89, 0.85, 0.78 and 0.87, respectively. It is important to note that compared to ε equal to 0.4, the matching rate increased by 11%; on the other hand, the number of passengers with waiting time equal to 0 decreased from 163783 to 145980.

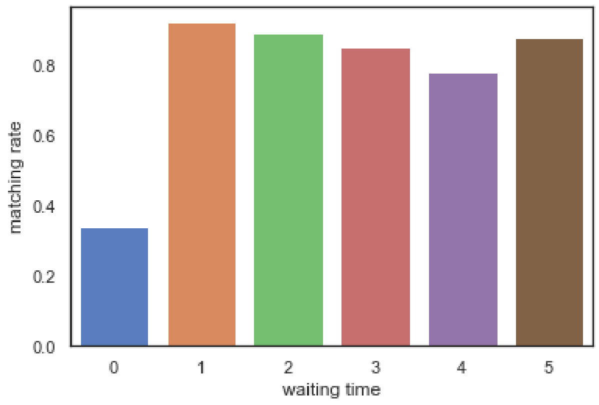


FIGURE 5.21: Matching rate on waiting time for $\varepsilon = 0.5$

This phenomenon suggests that an increase in the value of ε leads to longer scheduling flexibility, which can be divided between detour from the shortest path and waiting time. As a result, by increasing ε, the number of passengers with waiting time equal to 0 decreases from 195826 ($\varepsilon = 0.3$) to 145980 ($\varepsilon = 0.5$). At the same time, the number of matched passengers for waiting time equal to 0 increased from 38165 ($\varepsilon = 0.3$) to 49330 ($\varepsilon = 0.5$).

In figure 5.22, the hourly matching rate is shown, and the pattern for the matching rate is similar to the hourly matching rate for $\varepsilon = 0.5$. The maximum matching rate is 0.86 at 19 and is followed by two peaks at 22 and 23 by 0.82 and 0.79 matching rate, respectively. The minimum matching rates occur at 5, and 6 with 0.24 and 0.23 of the passengers getting matched. The number of matched passengers increased from 234992 to 267748. While, at 5 and 6, the number of matched passengers only increased by 16, which shows the importance of critical mass in the formation of sharing economy.

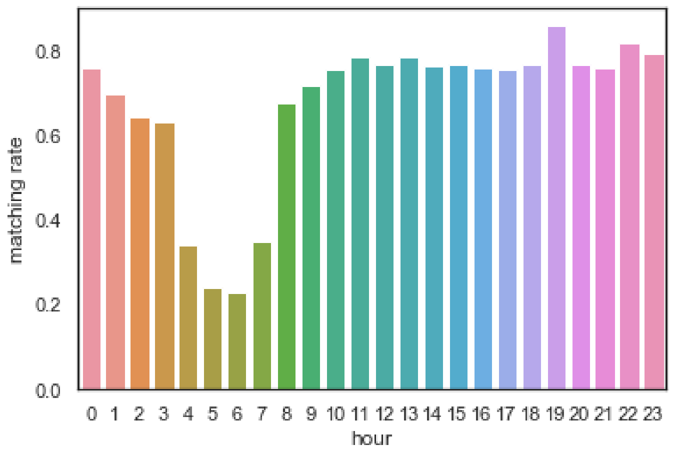


FIGURE 5.22: Hourly matching rate for $\varepsilon = 0.6$

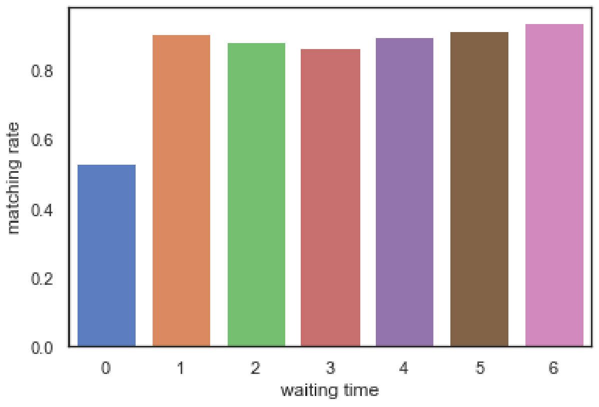


FIGURE 5.23: Matching rate on waiting time for $\varepsilon = 0.6$

The matching rate based on the waiting time for $\varepsilon = 0.6$ is shown in figure 5.23. For waiting time equal to 0, the matching rate increases to 53%, while reaching to the lowest matching rate among different waiting times. It is important to note that even for large scheduling flexibility ($\varepsilon = 0.6$), 47% of the passengers with waiting time equal to 0 cannot find a match and therefore accomplish their trip alone. The maximum matching rate is for waiting time equal to 6, where 3352 requests out of 3565 find a match and share their ride with a passenger.

The hourly matching rate for $\varepsilon = 0.7$ is shown in figure 5.24. The matching rate for busy hours is around 80%, and we observe peak at 1, 9, 10, 19, 22 and, 23 with a matching rate equal to 84%, 86%, 90%, 90%, 86%, and 84%, respectively. It is important to note that the matching procedure is saturated for unbusy hours (5, 6), where the number of matched passengers increased by 8 by increasing ε from 0.6 to 0.7. At 4 and 7, we observe the same pattern as 5 and 6; the matching rate for these hours started at 32% and 31% for $\varepsilon = 0.4$. It monotonically increased up to 36% and 37% for $\varepsilon = 0.7$.

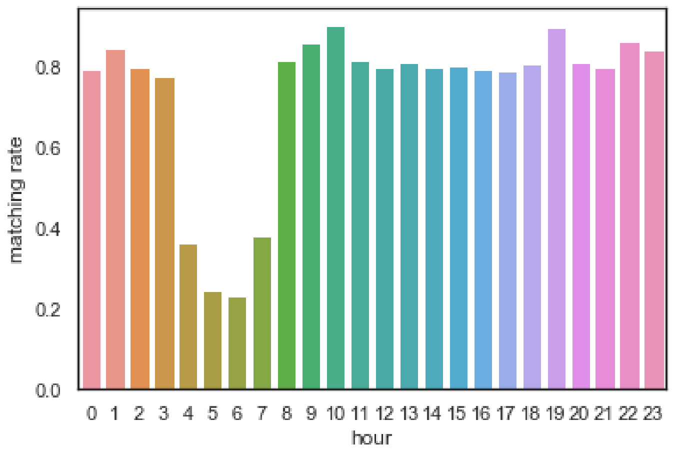


FIGURE 5.24: Hourly matching rate for $\varepsilon = 0.7$

In figure 5.25, the matching rate is plotted for values of the waiting time, which varies between 0 to 7. The matching rate for passengers with waiting time equal to zero is 52%. The matching rate for the passengers with waiting time equal to 5 is 97%, which is followed by 95% and 94.9% for waiting time equal to 6 and 4, respectively. It is important to note that the matching rate among passengers with waiting time greater than 0 is more than 93%, which indicates the ability of the waiting time function to increase the number of the matched passengers by defining the exact moment in which they should be matched.

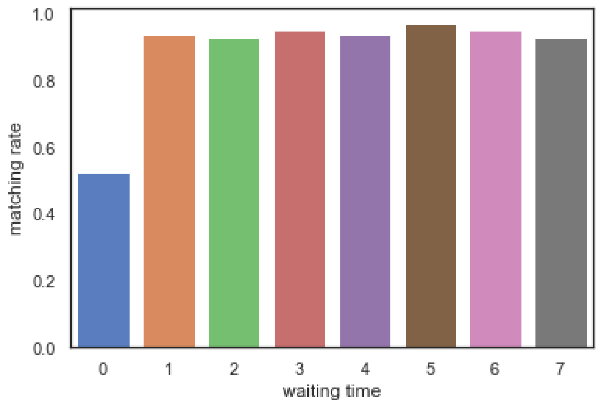


FIGURE 5.25: Matching rate on waiting time for $\varepsilon = 0.7$

Before providing the consequences of matching procedure on passenger trips (mean of increase in travel duration, mean of increase in pick up time) and also environmental consequences (the reduction in the total traveled distance), we provide the hourly matching rate and matching rate based on waiting time for $\varepsilon = 0.8$. In figure 5.26, the hourly matching rate is shown, and we observe the same pattern as the previous figure for the hourly matching rate. The overall matching rate reached 86% on $\varepsilon = 0.8$, and 306204 out of 356049 passengers found a match, and only 49845 passengers accomplished their rides alone.

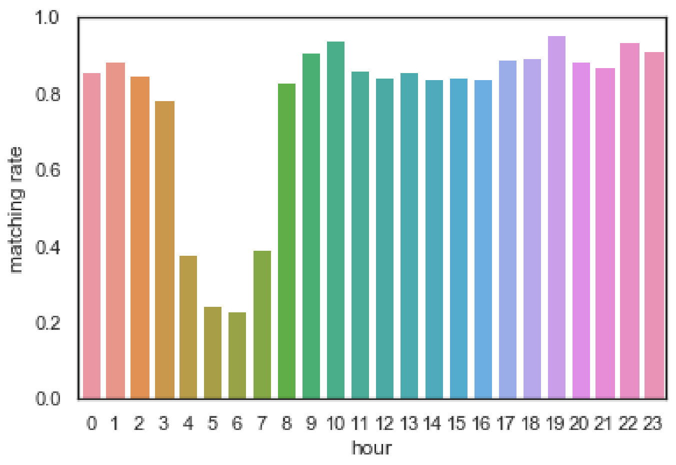


FIGURE 5.26: Hourly matching rate for $\varepsilon = 0.8$

In figure 5.27, we show the matching rate for different values of waiting time. When $\varepsilon = 0.8$, the passengers' waiting time can vary between 0 to 8. For ε equal to 0.8 which means that the scheduling flexibility of the passengers in equal to 80% of the shortest path, the matching rate for waiting time equal to 0 is 52% and 48% of the passengers with waiting time equal to 0 need to accomplish their ride alone.

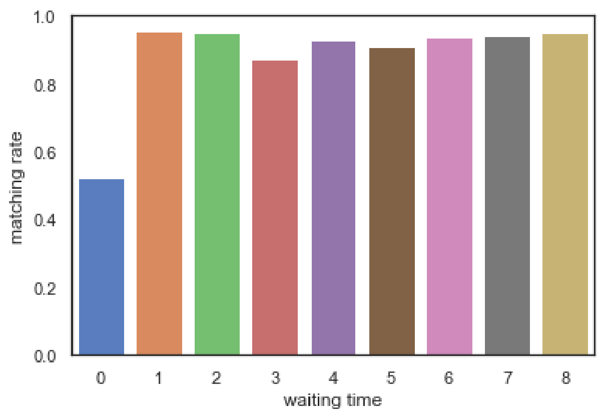


FIGURE 5.27: Matching rate on waiting time for $\varepsilon = 0.8$

Our observation shows that passengers with short travel time (less than 5 minutes) are hard to match and obtain a waiting time equal to zero, specifically on the unbusy hours. On the other hand, passengers with relatively long trip duration (more than 15 minutes) are easily matched and obtain larger value for their waiting time.

Here we provide the results of our experiments on the overall passenger experience. To measure the trip experience for passengers, we calculate the mean of waiting time, the mean of pickup, the mean of travel overhead, and total reduction in trip cost using ridesharing. Besides that, to observe the environmental contribution of ridesharing, the reduction in total traveled distance is calculated and reported based on the value of ε. It is important to note that in the case in which two passengers are allowed to share their ride, even if their pickup and dropoff locations are identical, the actual travel path is reduced by 50% and is the upper bound for the reduction in the actual travel path.

ε	Matching rate	Mean of waiting (in minutes)	Mean of pickup (in minutes)	Mean of travel overhead (in minutes)
0.3	0.415	0.5964	0.9084	2.4384
0.4	0.538	0.8843	1.1703	3.0603
0.5	0.66	1.1887	1.6187	3.5787
0.6	0.752	1.5585	2.1715	4.334
0.7	0.806	1.93	2.724	5.478
0.8	0.86	2.234	3.162	6.157

TABLE 5.2: Trips characteristics based on ε

In table 5.2, the trip characteristics are provided. The mean of waiting represents the mean of waiting time for all the passengers. The mean waiting time is started at 0.5964 for $\varepsilon = 0.3$ and then increased to 2.234 for $\varepsilon = 0.8$. The mean of pickup represents the mean of overall waiting before vehicle arrival (including waiting time and pickup delay). It is started by 0.9084 for $\varepsilon = 0.3$ and increased to 3.162 for $\varepsilon = 0.8$. Having the mean travel overhead, we can calculate the mean of difference between traveling on the shortest path and traveling using ridesharing. Travel overhead includes waiting time, pickup delay, and detour from the shortest path. It is started at 2.4384 for $\varepsilon = 0.3$ and increased to 6.157 for $\varepsilon = 0.8$. In our simulation, 41% of the passengers can share their ride by increasing the duration of their trips by the average of 2.4384 minutes, and by increasing ε to 0.8, 86% of the passengers can share their ride by extending their trip duration by 6.157 minutes on average.

ε	Reduction in total traveled distance	Reduction in Total cost
0.3	0.045	0.038
0.4	0.069	0.0573
0.5	0.1122	0.09826
0.6	0.1429	0.1211
0.7	0.18135	0.1533
0.8	0.215	0.18732

TABLE 5.3: Reduction in total traveled distance and total cost

When two people accomplish a trip together, their trip cost is split between them equally and must be less than or equal to the case that they travel alone. In table 5.3, the reduction in total traveled distance, and the reduction in total cost are calculated based on ε, which varies between 0.3 to 0.8. In the case that passengers travel alone, the total traveled distance is 615771.23 km. For $\varepsilon = 0.3$, the total traveled distance reduces by 4.5%, meaning that it is 27709 km less, and the total reduction in cost is 3.8%. By increasing the ε to 0.4, the total traveled distance reduces by 6.9% and causes saving 42488 km in the total traveled distance while the total cost is reduced by 5.73%. For $\varepsilon = 0.5$, with increasing 3.5787 minutes in trip duration, 11.22% of the total traveled distance is saved (69089.5 km). It leads to a reduction in the total cost by 9.826%. For ε equal to 0.6, the total traveled distance is reduced by 14.29% (87993.7 km), while it reduces the total cost by 12.11%. By increasing ε to 0.7, the total traveled distance is reduced by 18.135% (111670.1 km), and the total cost is reduced by 15.33%. For $\varepsilon = 0.8$, the total traveled distance is reduced by 21.5% (132390.8 km), decreasing the total cost by 18.732%.

Chapter 6

Conclusion and Future Work

Sharing is a basic behavior in humankind, and it dominated the hunter-gatherer societies. As the size of societies increased, the sharing behavior narrowed to intimate societies like families, neighbors, and friends. During the information revolution and by the creation of cyberspace, the modern human found a new opportunity for their social experience. During the past 30 years, individuals have formed a new form of identity known as online identity, leading to a safe communication within individuals in cyberspace. By the emergence of smartphones in the late 2000s and inclusion of internet, GPS, and other technologies in smartphones, the sharing became popular to conduct sharing on the cyberspace according to the online identity. During the past 10 years, the sharing economy has gained lots of attention. It has been implemented in different markets, from lodging in Airbnb to transportations like Uber, Lyft, and Didi.

After the industrial revolution, the human population moved from rural areas to urban areas, and for the first time in 2007, the urban population exceeded the rural population [76]. In the urbanized area with a high population density, people created a massive dynamic of movements based on their transportation needs. Using vehicles was one of the dominant transportation modes. This popularity of using vehicles has led to heavy congestion during the rush hours in the road network, which results in individual, social, and environmental costs. In 2017, 8.8 billion hours were wasted in congestion just in the United States, and besides that, it caused 3.3 billion gallons of fuel wasted [83]. Human consumption behavior is shifting towards environmental consciousness and sustainability, which leads to considering sharing as an alternative solution for owning; the same pattern of behavior occurs in transportation. The efficient use of empty seats in vehicles can reduce the number of vehicles in the road network, which is beneficial on individual, social and environmental levels.

The concept of sharing a ride started in the United States in an offline manner during World War II to reduce the consumption of resources. The ridesharing concept remained marginalized until the 21st century when it initially began to mature by the emergence of necessary technologies in smartphones for the implementation of online ridesharing. In online ridesharing, the decision-maker needs to respond in a short time, usually in a few minutes, to handle requests in an online manner. In this thesis, we extensively review the state-of-the-art works in ridesharing. We introduced our ridesharing system in the case that passengers are willing to increase their trip time by a fixed flexibility factor (ε). We also introduced the concept of adaptive waiting time, which divides the increased trip duration to waiting time and detour from the shortest path in order to minimize the passenger expected cost with respect to ε and spatial and temporal characteristics of the request.

Next, we designed a simulation to observe the performance of our ridesharing approach in practice. The simulation environment is a rectangle with an area of 44 km^2, and we considered all the request comes and goes to this rectangle in NYC on January 26th, which includes 356049 requests. The data is streamed in one-minute intervals (1440 slots for a day), and at each slot, a set of new requests were revealed. The optimal waiting time is then calculated based on future demand, and passengers were added to the passengers pool to be matched when their waiting time elapsed. The simulation is repeated for different values of ε varied between 0.3 and 0.8, and we provide the results in a mean of trip characteristic. Based on our simulation results for epsilon equal to 0.6, 75.2% of the passengers can share their ride if their trip duration was increased 4.334 minutes on average. It leads to a reduction in the overall cost by 12%. It also reduces the vehicles' overall traveled distance by 14.29%. Our observation shows that the adaptive waiting time concept makes the passengers' graph dense, and even with a naive matching procedure, it is able to match up to 86% of the passengers. One way to enhance the reduction, both in the total traveled distance and in total cost, is to use a non-greedy matching algorithm that could use the passenger graph efficiently.

The future direction for ridesharing is vast, and the authors proposed to investigate the following area for research:

1. Expanding the k-server problem for the ridesharing setting:

    - In the *Uber*-problem, a request has a pickup and dropoff, and a server needs to move from pickup to dropoff.

- In *Serversharing*-problem setting, each request is a set of pickup and dropoff locations, and the server needs to move based on their order to serve all the requests in a set (matched passenger).

2. Find the optimal policy for vehicles routing using reinforcement learning:

   - Using reinforcement learning can lead to finding the optimal policy in order to satisfy the requests and minimize the vehicles' total traveled distance.

3. Developing machine learning technique to solve minimum weight maximal matching:

   - The problem is $\mathcal{NP}$, and finding the exact solution is computationally intractable.
   - The graph-based machine learning approach can lead to finding a competitive solution in a short time.

4. Developing machine learning models for pickup-dropoff prediction:

   - Massive data set with millions of ride requests available for training potent machine learning models.
   - Integrating future requests in decision making can lead to a better solution and has recently achieved much attraction from the ML community.

5. Using the adaptive waiting time as the insertion process in the online algorithm for ridesharing.

Bibliography

[1] Niels A.H. Agatz et al. “Dynamic ride-sharing: A simulation study in metro Atlanta”. In: *Transportation Research Part B: Methodological* 45.9 (2011). Select Papers from the 19th ISTTT, pp. 1450–1464. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2011.05.017. URL: http://www.sciencedirect.com/science/article/pii/S0191261511000671.

[2] Niels Agatz et al. “Optimization for dynamic ride-sharing: A review”. In: *European Journal of Operational Research* 223.2 (2012), pp. 295–303. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.05.028. URL: http://www.sciencedirect.com/science/article/pii/S0377221712003864.

[3] Javier Alonso-Mora et al. “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment”. In: *Proceedings of the National Academy of Sciences* 114.3 (2017), pp. 462–467. ISSN: 0027-8424. DOI: 10.1073/pnas.1611675114. eprint: http://www.pnas.org/content/114/3/462.full.pdf. URL: http://www.pnas.org/content/114/3/462.

[4] Andrew Amey, John Attanucci, and Rabi Mishalani. “Real-Time Ridesharing: Opportunities and Challenges in Using Mobile Phone Technology to Improve Rideshare Services”. In: *Transportation Research Record* 2217.1 (2011), pp. 103–110. DOI: 10.3141/2217-13. eprint: https://doi.org/10.3141/2217-13. URL: https://doi.org/10.3141/2217-13.

[5] Ahmed Atahran, Christophe Lenté, and Vincent T’kindt. “A Multicriteria Dial-a-Ride Problem with an Ecological Measure and Heterogeneous Vehicles”. In: *Journal of Multi-Criteria Decision Analysis* 21.5-6 (2014), pp. 279–298. DOI: 10.1002/mcda.1518. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mcda.1518. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mcda.1518.

[6] Maria Battarra, Jean-François Cordeau, and Manuel Iori. "Chapter 6: Pickup-and-Delivery Problems for Goods Transportation". In: *Vehicle Routing*, pp. 161–191. DOI: 10.1137/1.9781611973594.ch6. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611973594.ch6. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611973594.ch6.

[7] Russell Belk. "Sharing". In: *Journal of Consumer Research* 36.5 (Aug. 2009), pp. 715–734. ISSN: 0093-5301. DOI: 10.1086/612649. eprint: https://academic.oup.com/jcr/article-pdf/36/5/715/5053101/36-5-715.pdf. URL: https://doi.org/10.1086/612649.

[8] Russell Belk. "Why Not Share Rather Than Own?" In: *The ANNALS of the American Academy of Political and Social Science* 611.1 (2007), pp. 126–140. DOI: 10.1177/0002716206298483. eprint: https://doi.org/10.1177/0002716206298483. URL: https://doi.org/10.1177/0002716206298483.

[9] Irwan Bello et al. "Neural Combinatorial Optimization with Reinforcement Learning". In: *CoRR* abs/1611.09940 (2016). arXiv: 1611.09940. URL: http://arxiv.org/abs/1611.09940.

[10] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon". In: *CoRR* abs/1811.06128 (2018). arXiv: 1811.06128. URL: http://arxiv.org/abs/1811.06128.

[11] Russell W. Bent and Pascal Van Hentenryck. "Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers". In: *Operations Research* 52.6 (2004), pp. 977–987. DOI: 10.1287/opre.1040.0124. eprint: https://doi.org/10.1287/opre.1040.0124. URL: https://doi.org/10.1287/opre.1040.0124.

[12] Lawrence Bodin and Thomas R. Sexton. "The multi-vehicle subscriber dial-a-ride problem". In: 1986.

[13] Karin Bradley and Daniel Pargman. "The sharing economy as the commons of the 21st century". In: *Cambridge Journal of Regions, Economy and Society* 10 (July 2017), pp. 231–247. DOI: 10.1093/cjres/rsx001.

[14] Leo Breiman. "Arcing classifier (with discussion and a rejoinder by the author)". In: *Ann. Statist.* 26.3 (June 1998), pp. 801–849. DOI: 10.1214/aos/1024691079. URL: https://doi.org/10.1214/aos/1024691079.

[15] Leo Breiman. *Classification and Regression Trees*. en. Routledge, Oct. 2017. ISBN: 9781315139470. DOI: 10.1201/9781315139470. URL: https://www.taylorfrancis.com/books/9781315139470.

[16] Bert Brunekreef and Stephen T Holgate. “Air pollution and health”. In: *The Lancet* 360.9341 (2002), pp. 1233–1242. ISSN: 0140-6736. DOI: https://doi.org/10.1016/S0140-6736(02)11274-8. URL: http://www.sciencedirect.com/science/article/pii/S0140673602112748.

[17] Roberto Calvo and Alberto Colorni. “An effective and fast heuristic for the Dial-a-Ride problem”. In: *4OR* 5 (Apr. 2007), pp. 61–73. DOI: 10.1007/s10288-006-0018-0.

[18] Rich Caruana and Alexandru Niculescu-Mizil. “An Empirical Comparison of Supervised Learning Algorithms”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 161–168. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143865. URL: http://doi.acm.org/10.1145/1143844.1143865.

[19] Rabindra Nath Chakraborty. “Sharing Culture and Resource Conservation in Hunter-Gatherer Societies”. In: *Oxford Economic Papers* 59.1 (2007), pp. 63–88. ISSN: 00307653, 14643812. URL: http://www.jstor.org/stable/4500088.

[20] Nelson D. Chan and Susan A. Shaheen. “Ridesharing in North America: Past, Present, and Future”. In: *Transport Reviews* 32.1 (2012), pp. 93–112. DOI: 10.1080/01441647.2011.621557. eprint: https://doi.org/10.1080/01441647.2011.621557. URL: https://doi.org/10.1080/01441647.2011.621557.

[21] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[22] Jean-François Cordeau and Gilbert Laporte. “A tabu search heuristic for the static multi-vehicle dial-a-ride problem”. In: *Transportation Research Part B: Methodological* 37.6 (2003), pp. 579–594. ISSN: 0191-2615. DOI: https://doi.org/10.1016/S0191-2615(02)00045-0. URL: http://www.sciencedirect.com/science/article/pii/S0191261502000450.

[23] Jean-François Cordeau and Gilbert Laporte. "The dial-a-ride problem (DARP): Models and algorithms". In: *Annals OR* 153 (June 2007), pp. 29–46. DOI: 10.1007/s10479-007-0170-8.

[24] Jean-François Cordeau and Gilbert Laporte. "The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms". In: *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1.2 (June 2003), pp. 89–101. ISSN: 1619-4500. DOI: 10.1007/s10288-002-0009-8. URL: https://doi.org/10.1007/s10288-002-0009-8.

[25] Hanjun Dai et al. "Learning Combinatorial Optimization Algorithms over Graphs". In: *CoRR* abs/1704.01665 (2017). arXiv: 1704.01665. URL: http://arxiv.org/abs/1704.01665.

[26] Sina Dehghani et al. *Stochastic k-Server: How Should Uber Work?* 2017. arXiv: 1705.05755 [cs.DS].

[27] Jacques Desrosiers, Yvan Dumas, and François Soumis. "A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-A-Ride Problem with Time Windows". In: *American Journal of Mathematical and Management Sciences* 6.3-4 (1986), pp. 301–325. DOI: 10.1080/01966324.1986.10737198. eprint: https://doi.org/10.1080/01966324.1986.10737198. URL: https://doi.org/10.1080/01966324.1986.10737198.

[28] Tawanna R. Dillahunt and Amelia R. Malone. "The Promise of the Sharing Economy among Disadvantaged Communities". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: Association for Computing Machinery, 2015, pp. 2285–2294. ISBN: 9781450331456. DOI: 10.1145/2702123.2702189. URL: https://doi.org/10.1145/2702123.2702189.

[29] Y. Dumas, Jacques Desrosiers, and F. Soumis. "Large scale multi-vehicle dial-a-ride problems". In: (Jan. 1989).

[30] P. Ehsani and J. Y. Yu. "The merits of sharing a ride". In: *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2017, pp. 776–782.

[31] Alan Page Fiske. *Structures of Social Life: The Four Elementary Forms of Human Relations*. New York, NY, US: Free Press, 1991.

[32] Joshua Fogel and Elham Nehmad. “Internet social network communities: Risk taking, trust, and privacy concerns”. In: *Computers in Human Behavior* 25 (Jan. 2009), pp. 153–160. DOI: 10.1016/j.chb.2008.08.006.

[33] M. Fortun and S. S. Schweber. “Scientists and the Legacy of World War II: The Case of Operations Research (OR)”. In: *Social Studies of Science* 23.4 (1993), pp. 595–642. ISSN: 03063127. URL: http://www.jstor.org/stable/285727.

[34] Yoav Freund and Robert E. Schapire. “Experiments with a New Boosting Algorithm”. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML’96. Bari, Italy: Morgan Kaufmann Publishers Inc., 1996, pp. 148–156. ISBN: 1-55860-419-7. URL: http://dl.acm.org/citation.cfm?id=3091696.3091715.

[35] Jerome H. Friedman. “Greedy function approximation: A gradient boosting machine.” In: *Ann. Statist.* 29.5 (Oct. 2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451. URL: https://doi.org/10.1214/aos/1013203451.

[36] SARAH FROHARDT-LANE. “Promoting a Culture of Driving: Rationing, Car Sharing, and Propaganda in World War II”. In: *Journal of American Studies* 46.2 (2012), pp. 337–355. URL: http://www.jstor.org/stable/23259140.

[37] Masabumi Furuhata et al. “Ridesharing: The state-of-the-art and future directions”. In: *Transportation Research Part B: Methodological* 57 (2013), pp. 28–46. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2013.08.012. URL: http://www.sciencedirect.com/science/article/pii/S0191261513001483.

[38] Thierry Garaix et al. “Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation”. In: *Computers Operations Research* 38.10 (2011), pp. 1435–1442. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2010.12.014. URL: http://www.sciencedirect.com/science/article/pii/S0305054810003102.

[39] Eleonora Gargiulo et al. “Dynamic Ride Sharing Service: Are Users Ready to Adopt it?” In: *Procedia Manufacturing* 3 (2015). 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015, pp. 777–784. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2015.07.329. URL: http://www.sciencedirect.com/science/article/pii/S2351978915003303.

[40] Robert Geisberger et al. *Fast Detour Computation for Ride Sharing*. 2009. arXiv: 0907.5269 [cs.DS].

[41] Kees Klein Goldewijk, Arthur Beusen, and Peter Janssen. “Long-term dynamic modeling of global population and built-up area in a spatially explicit way: HYDE 3.1”. In: *The Holocene* 20.4 (2010), pp. 565–573. DOI: 10.1177/0959683609356587. eprint: https://doi.org/10.1177/0959683609356587. URL: https://doi.org/10.1177/0959683609356587.

[42] Garrett Hardin. “The Tragedy of the Commons”. In: *Science* 162.3859 (1968), pp. 1243–1248. ISSN: 0036-8075. DOI: 10.1126/science.162.3859.1243. eprint: https://science.sciencemag.org/content/162/3859/1243.full.pdf. URL: https://science.sciencemag.org/content/162/3859/1243.

[43] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[44] Kristen Hawkes. “Showing off: Tests of an hypothesis about men’s foraging goals”. In: *Ethology and Sociobiology* 12.1 (1991), pp. 29–54. ISSN: 0162-3095. DOI: https://doi.org/10.1016/0162-3095(91)90011-E. URL: http://www.sciencedirect.com/science/article/pii/016230959190011E.

[45] Suining He and Kang G. Shin. “Spatio-Temporal Capsule-Based Reinforcement Learning for Mobility-on-Demand Network Coordination”. In: *The World Wide Web Conference*. WWW ’19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 2806–2813. ISBN: 9781450366748. DOI: 10.1145/3308558.3313401. URL: https://doi.org/10.1145/3308558.3313401.

[46] David A. Hensher. “Climate change, enhanced greenhouse gas emissions and passenger transport – What can we do to make a difference?” In: *Transportation Research Part D: Transport and Environment* 13.2 (2008), pp. 95–111. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2007.12.003. URL: http://www.sciencedirect.com/science/article/pii/S1361920907001320.

[47] Sin C. Ho et al. “A survey of dial-a-ride problems: Literature review and recent developments”. In: *Transportation Research Part B: Methodological* 111 (2018), pp. 395–421. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2018.

02.001. URL: http://www.sciencedirect.com/science/article/pii/S0191261517304484.

[48] John Holler et al. *Deep Reinforcement Learning for Multi-Driver Vehicle Dispatching and Repositioning Problem*. 2019. arXiv: 1911.11260 [cs.LG].

[49] Joel L. Horowitz. "Modifying urban transportation systems to improve the urban environment". In: *Computers Operations Research* 3.2 (1976), pp. 175–183. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(76)90027-7. URL: http://www.sciencedirect.com/science/article/pii/0305054876900277.

[50] Laurent Hyafil and Ronald L. Rivest. "Constructing optimal binary decision trees is NP-complete". In: *Information Processing Letters* 5.1 (1976), pp. 15–17. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(76)90095-8. URL: http://www.sciencedirect.com/science/article/pii/0020019076900958.

[51] Esa Hyytiä, Aleksi Penttinen, and Reijo Sulonen. "Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives". In: *Computers Operations Research* 39.12 (2012), pp. 3021–3030. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2012.03.002. URL: http://www.sciencedirect.com/science/article/pii/S030505481200055X.

[52] Jang-Jei Jaw et al. "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows". In: *Transportation Research Part B: Methodological* 20.3 (1986), pp. 243–257. ISSN: 0191-2615. DOI: https://doi.org/10.1016/0191-2615(86)90020-2. URL: http://www.sciencedirect.com/science/article/pii/0191261586900202.

[53] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *NIPS*. 2017.

[54] Robert Laurens Kelly. *The Foraging Spectrum: Diversity in Hunter-Gatherer Lifeways*. Smithsonian Institution Press, 1995.

[55] Steven Kuhn. "Prisoner's Dilemma". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2019. Metaphysics Research Lab, Stanford University, 2019.

[56] Nicholas D Kullman et al. "Dynamic Ridehailing with Electric Vehicles". working paper or preprint. Jan. 2020. URL: https://hal.archives-ouvertes.fr/hal-02463422.

[57] Michael W. Levin et al. “A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application”. In: *Computers, Environment and Urban Systems* 64 (2017), pp. 373–383. ISSN: 0198-9715. DOI: `https://doi.org/10.1016/j.compenvurbsys.2017.04.006`. URL: `http://www.sciencedirect.com/science/article/pii/S019897151630237X`.

[58] Xihan Li et al. “A Cooperative Multi-Agent Reinforcement Learning Framework for Resource Balancing in Complex Logistics Network”. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’19. Montreal QC, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 980–988. ISBN: 9781450363099.

[59] Kaixiang Lin et al. “Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. KDD ’18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 1774–1783. ISBN: 9781450355520. DOI: `10.1145/3219819.3219993`. URL: `https://doi.org/10.1145/3219819.3219993`.

[60] L. Liu et al. “Contextualized Spatial–Temporal Network for Taxi Origin-Destination Demand Prediction”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.10 (2019), pp. 3875–3887.

[61] Mustafa Lokhandwala and Hua Cai. “Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of NYC”. In: *Transportation Research Part C: Emerging Technologies* 97 (2018), pp. 45–60. ISSN: 0968-090X. DOI: `https://doi.org/10.1016/j.trc.2018.10.007`. URL: `http://www.sciencedirect.com/science/article/pii/S0968090X18307551`.

[62] S. Ma, Y. Zheng, and O. Wolfson. “T-share: A large-scale dynamic taxi ridesharing service”. In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 2013, pp. 410–421.

[63] Chao Mao, Yulin Liu, and Zuo-Jun (Max) Shen. “Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach”. In: *Transportation Research Part C: Emerging Technologies* 115 (2020), p. 102626. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.102626. URL: http://www.sciencedirect.com/science/article/pii/S0968090X19312227.

[64] Abood Mourad, Jakob Puchinger, and Chengbin Chu. “A survey of models and algorithms for optimizing shared mobility”. In: *Transportation Research Part B: Methodological* 123 (2019), pp. 323–346. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2019.02.003. URL: http://www.sciencedirect.com/science/article/pii/S0191261518304776.

[65] Elinor Ostrom. *Governing the commons: The evolution of institutions for collective action*. Cambridge University Press, 1990.

[66] Victor Pimenta et al. “Models and algorithms for reliability-oriented Dial-a-Ride with autonomous electric vehicles”. In: *European Journal of Operational Research* 257.2 (2017), pp. 601–613. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2016.07.037. URL: http://www.sciencedirect.com/science/article/pii/S0377221716305744.

[67] Warren B. Powell. “A Stochastic Formulation of the Dynamic Assignment Problem, with an Application to Truckload Motor Carriers”. In: *Transportation Science* 30.3 (1996), pp. 195–219. DOI: 10.1287/trsc.30.3.195. eprint: https://doi.org/10.1287/trsc.30.3.195. URL: https://doi.org/10.1287/trsc.30.3.195.

[68] Lew Pratsch. “CARPOOLS: THE UNDERUTILIZED RESOURCE”. In: *American Society of Civil Engineers* (1974), pp. 49–52.

[69] Lew Pratsch. “Commuter Ridesharing”. In: *G.E.Gray and L.A. Hoel Public Transportation: Planing, operations, and management* (1979), pp. 168–187.

[70] John A. Price. “Sharing: The Integration of Intimate Economies”. In: *Anthropologica* 17.1 (1975), pp. 3–27. ISSN: 00035459. URL: http://www.jstor.org/stable/25604933.

[71] Harilaos N. Psaraftis. “A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem”. In: *Transportation Science* 14.2 (1980), pp. 130–154. DOI: 10.1287/trsc.14.2.130. eprint: https://doi.org/10.1287/trsc.14.2.130. URL: https://doi.org/10.1287/trsc.14.2.130.

[72] Harilaos N. Psaraftis. “An Exact Algorithm for the Single Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows”. In: *Transportation Science* 17.3 (1983), pp. 351–357. DOI: 10.1287/trsc.17.3.351. eprint: https://doi.org/10.1287/trsc.17.3.351. URL: https://doi.org/10.1287/trsc.17.3.351.

[73] Luca Quadrifoglio, Maged M. Dessouky, and Fernando Ordóñez. “Mobility Allowance Shuttle Transit (MAST) Services: MIP Formulation and Strengthening with Logic Constraints”. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Ed. by Laurent Perron and Michael A. Trick. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 387–391. ISBN: 978-3-540-68155-7.

[74] Brahim Rekiek, Alain Delchambre, and Hussain Aziz Saleh. “Handicapped Person Transportation: An application of the Grouping Genetic Algorithm”. In: *Engineering Applications of Artificial Intelligence* 19.5 (2006), pp. 511–520. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2005.12.013. URL: http://www.sciencedirect.com/science/article/pii/S0952197606000339.

[75] Jeremy Rifkin. *The Zero Marginal Cost Society: The Internet of Things, the Collaborative Commons, and the Eclipse of Capitalism.* 1st. St. Martin’s Press, 2014.

[76] Hannah Ritchie and Max Roser. “Urbanization”. In: *Our World in Data* (2020). https://ourworldindata.org/urbanization.

[77] Filipe Rodrigues, Ioulia Markou, and Francisco C. Pereira. “Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach”. In: *Information Fusion* 49 (2019), pp. 120–129. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2018.07.007. URL: http://www.sciencedirect.com/science/article/pii/S1566253517308175.

[78] A. Santos et al. *Summary of travel trends: 2009 national household travel survey*. Tech. rep. US Department of Transportation Federal Highway Administration, 2011.

[79] M. W. P. Savelsbergh and M. Sol. "The General Pickup and Delivery Problem". In: *Transportation Science* 29.1 (1995), pp. 17–29. DOI: 10.1287/trsc.29.1.17. eprint: https://doi.org/10.1287/trsc.29.1.17. URL: https://doi.org/10.1287/trsc.29.1.17.

[80] M. Schilde, K.F. Doerner, and R.F. Hartl. "Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem". In: *European Journal of Operational Research* 238.1 (2014), pp. 18–30. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2014.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0377221714002197.

[81] M. Schilde, K.F. Doerner, and R.F. Hartl. "Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports". In: *Computers Operations Research* 38.12 (2011), pp. 1719–1730. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2011.02.006. URL: http://www.sciencedirect.com/science/article/pii/S0305054811000475.

[82] Juliet Schor. "Debating the Sharing Economy". In: *Great Transition Initiative* (Oct. 2014).

[83] David Schrank, Bill Eisele, and Tim Lomax. *2019 URBAN MOBILITY REPORT*. Tech. rep. Texas AM University System ,Texas AM Transportation Institute, 2019.

[84] David Schrank, Bill Eisele, and Tim Lomax. *TTI's 2012 URBAN MOBILITY REPORT Powered by INRIX Traffic Data*. Tech. rep. Texas AM University System ,Texas AM Transportation Institute, 2012.

[85] Joseph P. Schwieterman. "Uber Economics: Evaluating the Monetary and Travel Time Trade-Offs of Transportation Network Companies and Transit Service in Chicago, Illinois". In: *Transportation Research Record* 2673.4 (2019), pp. 295–304. DOI: 10.1177/0361198119839344. eprint: https://doi.org/10.1177/0361198119839344. URL: https://doi.org/10.1177/0361198119839344.

[86] Zhenyu Shou and Xuan Di. *Reward Design for Driver Repositioning Using Multi-Agent Reinforcement Learning*. 2020. arXiv: 2002.06723 [cs.LG].

[87] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. arXiv: `1206.2944 [stat.ML]`.

[88] Robert Sparrow and Mark Howard. "When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport". In: *Transportation Research Part C: Emerging Technologies* 80 (2017), pp. 206–215. ISSN: 0968-090X. DOI: `https://doi.org/10.1016/j.trc.2017.04.014`. URL: `http://www.sciencedirect.com/science/article/pii/S0968090X17301262`.

[89] Niranjan Srinivas et al. "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting". In: *IEEE Transactions on Information Theory* 58.5 (May 2012), pp. 3250–3265. ISSN: 1557-9654. DOI: `10.1109/tit.2011.2182033`. URL: `http://dx.doi.org/10.1109/TIT.2011.2182033`.

[90] Mitja Stiglic et al. "Making dynamic ride-sharing work: The impact of driver and rider flexibility". In: *Transportation Research Part E: Logistics and Transportation Review* 91 (2016), pp. 190–207. ISSN: 1366-5545. DOI: `https://doi.org/10.1016/j.tre.2016.04.010`. URL: `http://www.sciencedirect.com/science/article/pii/S1366554515303033`.

[91] Mitja Stiglic et al. "The benefits of meeting points in ride-sharing systems". In: *Transportation Research Part B: Methodological* 82 (2015), pp. 36–53. ISSN: 0191-2615. DOI: `https://doi.org/10.1016/j.trb.2015.07.025`. URL: `http://www.sciencedirect.com/science/article/pii/S0191261515002088`.

[92] Z. Caner Taskin and Tinaz Ekim. "Integer programming formulations for the minimum weighted maximal matching problem". In: *Optimization Letters* 6.6 (Aug. 2012), pp. 1161–1171. ISSN: 1862-4480. DOI: `10.1007/s11590-011-0351-x`. URL: `https://doi.org/10.1007/s11590-011-0351-x`.

[93] New York (N. Y.). Taxi and Limousine Commission. "New York City Taxi Trip Data, 2009-2018". In: (2019). DOI: `10.3886/ICPSR37254.v1`.

[94] Tanvi Verma et al. *Augmenting Decisions of Taxi Drivers through Reinforcement Learning for Improving Revenues*. 2017. URL: `https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15746`.

[95] D. Wang et al. "DeepSD: Supply-Demand Prediction for Online Car-Hailing Services Using Deep Neural Networks". In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 2017, pp. 243–254.

[96] James Woodburn. “Egalitarian Societies”. In: *Man* 17.3 (1982), pp. 431–451. ISSN: 00251496. URL: `http://www.jstor.org/stable/2801707`.

[97] Zhihai Xiang, Chengbin Chu, and Haoxun Chen. “The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments”. In: *European Journal of Operational Research* 185.2 (2008), pp. 534–551. ISSN: 0377-2217. DOI: `https://doi.org/10.1016/j.ejor.2007.01.007`. URL: `http://www.sciencedirect.com/science/article/pii/S0377221707000926`.

[98] Xin Xing et al. “SMIZE: A Spontaneous Ride-Sharing System for Individual Urban Transit”. In: *Multiagent System Technologies*. Ed. by Lars Braubach et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 165–176. ISBN: 978-3-642-04143-3.

[99] J. Xu et al. “A Sequence Learning Model with Recurrent Neural Networks for Taxi Demand Prediction”. In: *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. 2017, pp. 261–268.

[100] Huaxiu Yao et al. “Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction”. In: *CoRR* abs/1802.08714 (2018). arXiv: `1802.08714`. URL: `http://arxiv.org/abs/1802.08714`.

[101] Seyed Amir H. Zahabi et al. “Urban Transportation Greenhouse Gas Emissions and Their Link with Urban Form, Transit Accessibility, and Emerging Green Technologies: Montreal, Quebec, Canada, Case Study”. In: *Transportation Research Record* 2375.1 (2013), pp. 45–54. DOI: `10.3141/2375-06`. eprint: `https://doi.org/10.3141/2375-06`. URL: `https://doi.org/10.3141/2375-06`.