UNIVERSITY OF CALIFORNIA,
IRVINE

An optimization Framework for Shared Mobility in Dynamic Transportation Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Civil Engineering

by

Neda Masoud

2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to express my sincere gratitudes to my ITS family who have made the past few years some of the best years of my life. The warm and supportive environment created by the faculty, staff, and students made ITS feel like home to me.

I am utterly grateful to my adviser, Professor R. Jayakrishnan, for giving me the chance to explore different directions until I found a subject I was passionate about, always making time for me when I needed someone to brainstorm with, generously sharing his invaluable advice and experience, patiently explaining the logic behind every single grammar correction, and last but not least, introducing me to Bibimbap!

My sincere gratitudes are due to my committee members Professor Will Recker, Professor Michael McNally, Professor Amelia Regan, and Professor John Turner for their endless support, never-ending encouraging words, insightful comments, and priceless advice.

I am indebted to Professor Stephen Ritchie for supporting me through the first two years of my Ph.D., and for being a part of my Ph.D. qualifying exam committee.

I would also like to thank Dr. Joseph Chow for his kind review of my work and his useful suggestions and advice, all the way from when he was my friend Joe whose office was a few doors away, to when he attained the very well-fitting and deserving “Professor” title.

I am also very appreciative of friends and colleagues at ITS who made the last five years memorable. Mahdieh, Mahboobeh, Nasrin, Mojtaba, Morteza, Amir, Mahdi, Sepehr, Fatemeh F., Akram, Anahita, Elnaz, Narges, Saeid, Maryam, Joe, Pedro, Christina, Fatemeh R., Robert, Daniel, Riju, Anupam, Kate, Ashley, Suman, Roger, Danny, and Gabe: thank you for making this journey an enjoyable one.

To my forever partner in crime, little sister Sara: thank you for always being there for me. Your positive attitude on life brings out the best in me.

I am utterly grateful to my parents for teaching me what matters in life, and giving me all the love and support a child could ask for; and, to my little brother Arshia, for putting his video games aside every once in a while to Skype with me!

A very special thanks goes to the most important person in my life: my kind, supportive, and beloved husband, Ali. He has always been there by my side, encouraged me to expand my horizons, and made me believe in myself by believing in me. Without him, this dissertation would not have been a possibility.

This work was financially supported by the U.S. Department of Transportation and the California Department of Transportation though the University of California Center on Economic Competitiveness in Transportation, the California Intelligent Transportation Systems scholarship, the Orange County Women in Transportation scholarship, and the Railway Association of Southern California scholarship. I am also very grateful to the Department of

Civil and Environmental Engineering, and the Institute of Transportation Studies at University of California, Irvine for their support.

CURRICULUM VITAE

Neda Masoud

EDUCATION

Doctor of Philosophy in Civil Engineering **2016**
University of California, Irvine *Irvine, CA*

Master of Science in Physics **2011**
University of Massachusetts Dartmouth *Dartmouth, MA*

Bachelor of Science in Industrial Engineering **2008**
Sharif University of Technology *Tehran, Iran*

ABSTRACT OF THE DISSERTATION

An optimization Framework for Shared Mobility in Dynamic Transportation Networks

By

Neda Masoud

Doctor of Philosophy in Civil Engineering

University of California, Irvine, 2016

Professor R. Jayakrishnan, Chair

Recent advances in communication technology coupled with increasing environmental concerns, road congestion, and the high cost of vehicle ownership have directed more attention to the opportunity cost of empty seats traveling throughout the transportation networks every day. Peer-to-peer (P2P) ridesharing is a good way of using the existing passenger-movement capacity on the vehicles, thereby addressing the concerns about the increasing demand for transportation that is too costly to address via infrastructural expansion.

This dissertation is dedicated to the optimization of the matching process between the participants in a ridesharing system. More specifically, focus of this dissertation is on multi-hop matching, in which riders have the possibility of transferring between vehicles. Different algorithms have been presented for various implementation strategies of ridesharing systems. Multiple case studies assess the important role ridesharing can play as a separate mode, or in conjunction with other modes of transportation, in multi-modal settings.

Chapter 1

Introduction

Recent advances in communication technology coupled with increasing environmental concerns, road congestion, and the high cost of vehicle ownership have directed more attention to the opportunity cost of empty seats traveling throughout the transportation networks every day. Peer-to-peer (P2P) ridesharing is a good way of using the existing passenger-movement capacity of vehicles, thereby addressing the concerns about the increasing demand for transportation that is too costly to address via infrastructural expansion.

Although limited versions of P2P ridesharing systems initially emerged in the US in the 1990s, they did not receive enough support from the targeted population to continue operating. Inadequate and non-targeted marketing, insufficient flexibility and convenience, and absence of appropriate technology were some of the factors that contributed to the lack of success in the implementation of the first generation of ridesharing systems.

A second generation of ridesharing systems emerged after considerable improvements in communications technology in the past few years. Making use of GPS-enabled cell phones in more recent ridesharing systems allows for accessing online information on the location of participants, making ridesharing more accessible, and providing people with a sense of

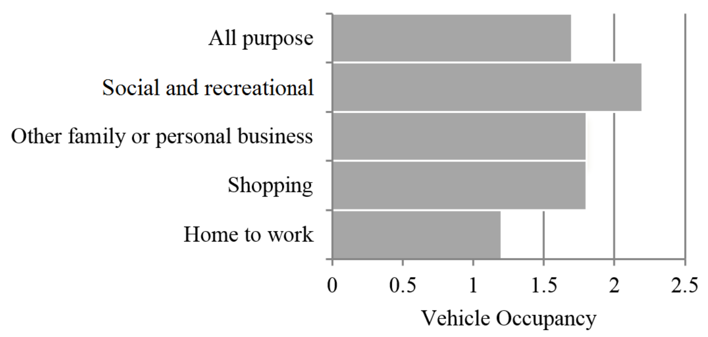


Figure 1.1: Average vehicle occupancy in the US in 2009 for different trip purposes (U.S. Department of Transportation, 2009)

security. These factors combined with the high cost of travel (both financial and environmental) have played an important role in the higher interest in ridesharing systems in the recent years. The high growth rate of Transportation Network Companies (TNC) such as Uber and Lyft in the USA and elsewhere in the world is an indicator of increasing levels of acceptance in the concept of outsourcing rides.

In line with the demand side, the supply side of P2P ridesharing has experienced growth as well. According to the 2009 national household travel survey (NHTS), out of an average of 4 seats available in a vehicle, only 1.7 is being actually used (Figure 1.1). This number is as low as 1.2 for work trips. In addition, number of trips per household in the US has experienced a decreasing trend since 1995. On the contrary, the general trend in the number of vehicles owned by households has been increasing. These statistics suggest that carpooling in households is declining, and that the number of empty seats available on traveling vehicles is increasing.

This increase on the supply side of ridesharing, coupled with the rise on the demand side imply an optimistic future for P2P ridesharing services. This potential was recognized by the US congress in June 2012. Section 1501 of the Moving Ahead for Progress in the 21^{st} Century (MAP-21) transportation act expanded the definition of “carpooling” to include “real-time ridesharing” as well, making ridesharing eligible for all the federal funds that were previously available only for carpooling projects. Ridesharing finds a mention in the Fixing America's Surface Transportation (FAST) Act of December 2015 as well, though proponents of ridesharing would be somewhat disappointed that there was no further expansion in the consideration of the concept.

In this dissertation, we define P2P dynamic ridesharing to include all one-time shared rides with any type of arrangement, whether it is on-the-fly or pre-arranged, between peer drivers and riders. Dynamic ridesharing differs from more traditional carpooling services in that in carpooling shared trips are scheduled for an extended period of time, and are not one time occurrences. Furthermore, the nature of carpooling programs does not ask for real-time ride-matching.

Drivers in a ridesharing system drive to perform activities of their own, and not for the mere purpose of transporting riders, hence the term "peer-to-peer". Each driver can have multiple riders on board at any point in time. In addition, to increase the number of satisfied ride requests, the system provides multi-hop itineraries for riders, i.e. riders may transfer between vehicles.

We define a set of stations in the network where riders and drivers can start and end their trips, and riders can switch between vehicles. The system finds matches for riders by optimally routing drivers in the network. In order to guarantee a high quality of service, both riders and drivers provide a time window to specify the start and end of their trip, and a maximum ride time. Furthermore, riders can specify the maximum number of transfers they are willing to make, and drivers can put a limit on the number of riders they want to have on board at each moment in time. The term "real-time" emphasizes the capability of the system to make ride-matches in a short period of time, for implementation with frequent re-optimizations using newer data over time.

A P2P ride-matching algorithm is central to the successful implementation of a ridesharing system. Ride-matching refers to the problem of matching riders (passengers) and drivers in a ridesharing system. A successfully matched rider receives from the system an itinerary of their trip that includes information on the scheduled route, and the drivers with whom the travel is planned. Drivers receive itineraries that include the schedules to pick up and drop off riders.

In the next Chapter, we provide a review of previous ridsharing projects, and the attempts made in academia to devise efficient matching algorithms. In Chapter 3, we elaborate on the architecture of the ridesharing system under study. Chapter 4 provides a mathematical formulation of the ride-matching problem. In Chapter 5, we propose a pre-processing procedure that makes this mathematical formulation computationally easier to solve, by reducing the size of its input sets. In Chapters 6-9, we propose different ride-matching algorithms for different implementation strategies in a ridesharing system. Finally, in Chapter 10 we look at the central role ridesharing can play with the emergence of driver-less vehicles.

Chapter 2

Literature Review

2.1 History of Ridesharing

P2P ridesharing systems emerged in the US in the 1990's, mostly under government-funded projects, and continue working throughout the world today. In this section some of the ridesharing projects from both early and current days are presented. The close study of these systems manifests their evolution through time. Figures 2.1 and 2.2 display a list of these projects, the details involved in them, and the lessons learnt by the experts that could be used in designing more successful ridesharing systems.

2.1.1 Bellevue Smart Traveler: Washington (Haselkorn et al. (1995))

Two Smart Traveler projects were piloted in Bellevue, a small city near Seattle in 1993 and 1995. Bellevue was selected as the location to pilot these projects due to the existence of a dense employment location in the city's downtown area, where the majority of commuters traveling in single occupancy vehicles (SOVs) caused severe congestion. Previous efforts to reduce the congestion using HOV lanes had not been successful. The purpose of the projects was to provide commuters with innovative communication technologies to help them arrange for HOV commuting to and from the employment center.

The 1993 project focused on examining the impact of cellular phones on the ridesharing experience and to see whether it would promote ridesharing. The project started in July 1992, and the demonstration phase ended in April 1994. The system consisted of a voice-activated matching service, which also provided real-time traffic conditions. At the end of the project, the researchers found that no one used cell phones for the purpose of ridesharing, suggesting that cell phones alone were not incentive enough for people to participate in ridesharing. In the participants' views, cell phones did not compensate for their perceived downsides of ridesharing such as lack of flexibility and convenience. The second phase of this project was piloted in 1995 built on the lessons learnt from the first phase, and lasted for 5 months. The focus of the second phase was on assessing the ridesharing demand. This time in addition to matching rides and real-time traffic conditions, the transit schedule was also provided to the users.

The results of this phase were more encouraging than the last phase. People generally preferred to offer rides than to accept them. In this phase there were 53 participants, and 148 ride requests, out of which only 6 shared rides happened. The necessity of more research on the psychology of people offering and accepting rides was one of the conclusions made the researchers. Towards the end of the project, a survey of the participants was conducted. Participants indicated that it was difficult to find matches, which the program coordinators believed was due to lack of enough number of participants. Participants also believed that searching and confirming matches was a time consuming process.

The program coordinators concluded that lack of knowledge was one of the reasons behind unpopularity of the program. They found that tangible incentives for using the program, and a guaranteed ride home could have positive impact on the participation rate. Also, people could be more confident if their ride was confirmed at least one hour before the departure time. Pre-screening the participants could also make people feel more secure to participate in the program. An additional conclusion was that to avoid confusion, rather than having people meet at arbitrary places, they should meet at pre-determined stations.

2.1.2 Los Angeles Smart Traveler (Giuliano et al. (1995))

This project was conducted in 1994 in the LA metropolitan area. Passengers could use 77 kiosks located throughout the region to access transit routes, traffic conditions, and find one-time or regular rides. The participants could call their matches and talk to them, or leave them a message.

The majority of the ride-matching requests were for regular, and not one-time, carpools. A future survey revealed that this was because people did not want to take or give rides to people they didn't know. The scope of the project was changed due to the Northridge earthquake, and no marketing was done to promote the program.

A survey conducted of 25 of the participants in the program showed that none of them used the system to find a one-time match. Three of the survey takers stated that they had been contacted by other people for a one-time carpool that never happened. The program coordinators found out that people preferred to turn to their family or social network for one-time rides. One of the reasons behind this preference was that people found communicating through voice messages and waiting for return phone calls frustrating. The program coordinators also believed that the low use of the system was due to lack of marketing.

The final recommendations of the project team was assessing the potential demand before investing in a costly system, ensuring the knowledge of the users of the system, and developing a ridesharing application.

2.1.3 Coachella Valley Transaction Network (Levofsky and Greenberg (2001))

This project was conducted in Riverside in southern California in 1994. Similar to the Los Angeles Smart Traveler project, this project provided traffic conditions, transit information, and one-time ridesharing matches via 4 kiosks located in areas with high retail and pedestrian activity. The project continued for 7 month, at the end of which it was decided that the costs were too high and the usage was too low for it to be continued. The high cost of the system was due to using expensive, market specific media in the kiosks. It was projected that such a system may become cost efficient in 6 year.

During the time the project was running, around 21,500 people accessed the kiosks. Only one third of the inquiries were related to ridesharing, and only 8% of the total 3,200 printouts were rideshare match lists. These user statistics suggesting the lack of interest in ridesharing contributed to the decision on not continuing the project.

2.1.4 Sacramento Dynamic Ridesharing project (Kowshik et al. (1993))

This project started in 1994. Out of the 360 people who participated in the program, only 10 requested a ride, and out of the 10 ride-matching requests only 1 was found. In general, the project was considered as an unsuccessful one, primarily due to security concerns, as surveys revealed. The surveys conducted at the end of the program suggested that a pre-screening process to assure safety, and a fixed payment scheme to avoid undesirable negotiations could have improved the program.

The program coordinators considered inadequate marketing as one of the reasons of the failure of the project. They also believe that HOV lanes alone were not incentive enough for people to engage in ridesharing. Finding ways to assure personal security of the participants was one of the main suggestions of the program coordinators.

2.1.5 Seattle Smart Traveler (Dailey et al. (1999))

This project was conducted in 1996, on the campus of university of Washington. This location was selected because it was assumed that students and faculty would be inclined to use dynamic ridesharing, mainly because of type of their schedules, their high level of computer literacy and access to communication technologies, and the limited number of parking spaces available on campus.

Users had to provide a phone number and an email address. The system sent a list of matches to the user's email address. The rider and driver met at pre-determined locations, which reduced confusion, but led to some degree of inflexibility at the same time.

The project started in March and ended in the November of 1996. During this period 2056 trips were registered, 3% of which were one-time trips. The project coordinates did not start marketing the program until the fall of 1996. After outreach to the target users, however, the number of one-time ride matches increased considerably, making the project coordinates believe that there exists a demand for dynamic ridesharing.

2.1.6 RideNow, California (Nelson Nygaard Consulting Associates and RideNow Inc (2006))

RideNow was a dynamic ridesharing pilot project whose demonstration phase started in November 2005. The program provided one-time ride matches to and from the Dublin/Pleasanton BART station. The goal of the project was to shift the demand from SOVs to transit by offering dynamic ridesharing, while preserving the convenience of solo driving for ridesharing users.

The system was designed to provide ”instant” matches for the users, minutes before they left home in the morning, and while they were in the BART train on their way back home in the afternoon. The program tried to put into practice the lessons learned from other ridesharing programs. They offered incentives, such as free BART tickets for registration, extra free tickets for making matches, and a guaranteed ride home. They also conducted a three-phase marketing program.

The implementation phase of the program started in November 2005, and the program terminated in May 2006. During this 6 month period a total of 121 people registered in the program. The registration started at around 20 in week 1, and took its pick through weeks 18-21 (from around 45 to 105) after a marketing campaign. 20% of the registrants preferred to drive, 22% preferred to ride, and the remaining 58% were indifferent.

Around 50% of the registered users actually used the system. During this 6 month period, 1179 ridesharing requests were made, of which 141 matches were made. The ratio of successful matches however increased after the marketing campaign attracted more registrants in March 2006. Before this marketing campaign, an average of 6 ride matches per week was made. This average jumped to around 22 per week in April.

Eventually, only around 8% of the requests ended up sharing rides. Although more ridesharing matches were made, people didn’t actually follow up and get together with their matches, especially on the way back home.

The program was considered costly and difficult to market, and hence didn’t go further than a pilot project. However, the program coordinators believed that it could be cost-efficient if offered as a part of a larger ridesharing program. A follow up on the participants revealed that the phone system was cumbersome to use for some participants. Also, participants preferred to have a longer notification time frame than the 15 minutes used by the program for the morning commute.

The program coordinates recommendations for future ridesharing programs include sustained marketing, offering parking spaces as an incentive where applicable, an easy-to-use phone system and more lead time for morning matches.

2.1.7 Goose Networks, San Francisco, California (Heinrich (2010))

The Goose Networks Co. developed a real-time dynamic ridesharing technology that used SMS texting as the communication tool. The system searched for the key works, such as ”today” or ”now” in sentences. After assuring of the proper functioning of the technology the corporation decided to assess the demand for such a service by conducting a pilot project at The Genentech Co., a biotechnology firm located in San Francisco.

The pilot project lasted 11 months, and 200 out of the 8000 employees at Genentech registered in the program. Despite Genentech's effort to support this program by extending their existing $4 per day commute incentive program to the ridesharers, only 50 matches were made and carpooled. The project coordinators found that this was partially due to the preference employees had in riding the well-equipped busses that Genentech offered. Also, participants had a lot of questions during different phases of the program and required a high level of customer service that was hard to sustain.

2.1.8 Avego, University Cork, Ireland (Heinrich (2010) and UCC (2013))

Avego started in 2007 in Ireland as the R&D division of the company Mapflow, and the company is now known as Carma. In 2009 Avego became a separate entity and first offered its real-time ridesharing service at the University College Cork (UCC), with the purpose of trying out the dynamic ridesharing technology. Avego's technology consisted of an iPhone app, coupled with internet access.

UCC had 20,000 students and 3,000 staff at the time. In this project 20 participants were selected and were each given a free iPhone with a subsidized phone contract in return for picking up and dropping off simulated riders (named "ghosts") at least 20 times per month. The payment was VMT-based, and micro-payments were transferred between participants' accounts.

The pilot program was launched in February 2011, from Carrigaline to Cork. This line was selected because a high proportion of workers came to Cork from Carrigaline, there was little public transport available between Carrigaline and Cork, and the main arterials entering the town were extremely congested.

As for security measures, after each match was made, each participant was given a PIN which was required to be provided to the match. In addition, each participant could rate the match based on the overall ridesharing experience. These ratings could be viewed by participants and were considered a tool to enhance security.

As incentive, the first 30 drivers signing up for the program received €100 toward purchase of a phone, and active drivers could also earn €40 each four month.

2.1.9 Go520, Washington (RTrip (2013) and O'Sullivan (2011))

The Go520 dynamic ridesharing project, conducted in the 14-mile SR520 corridor connecting Redmond and Seattle in Washington, is known as the world's largest real-time ridesharing pilot project to date. The first phase of the project started after Avego received a $400,000 grant from the Washington State DOT (WSDOT) in September 2010, and was launched in January 2011.

The SR520 corridor is a crowded corridor carrying 190,000 people (and 115,000 vehicles) on a daily basis. For the first phase of this project, the goal was to recruit 1000 people (250 drivers, and 750 riders) who pass the rigorous, state-mandated screening process.

The drivers were provided with the Avego driver real-time ridesharing app (for iPhone), and the riders could book a ride from any internet-enabled cell phone, from a PC, or through the Avego rider app.

The program offered several incentives to the participants: 1- all the participants received a starter pack including free gas cards, iPhone chargers, and detained information on the program. 1- drivers were paid a fee for offering their empty seats ($1 booking fee, plus 20 cents per VMT for each passenger). 2- drivers could earn up to $30 per month in free gas cards. 3- riders who used the program were given a $30 per month in Avego credits that could be used towards ridesharing payment. 4- after launching the program, in spring of 2011, a toll was placed for the vehicles who used the SR520 bridge, making carpooling a more attractive option.

The demonstration phase started in January 2011 with 30 participants. By the end of February, the number of participants had jumped up to 550, and by April, 962 people had registered in the system, close to the target number of participants for the first phase.

The success of this phase was indebted to the coverage by the media, Public Relations

(PR) campaigns and outreach events, and involvement of big employers such as Microsoft and University of Washington in the program. The number of participants in the program experienced considerable jumps after each outreach and marketing event. However, the most affective impacts were due to the support of employers within the area.

The program was impeded due to the mandatory and strict screening process required by the state. Participants were asked to provide their social security number and date of birth for criminal background checks, hold a liability insurance that covers at least $300,000 per accident, provide their driving record, and meet the vehicle maintenance guidelines. The pre-screening process although insuring safety and security of participates in the program, created a huge gap between the registration and approval which curbed the initial high interests. Also, many people felt that the amount of information they have to provide is not worth the service, despite the incentives.

The second phase of the project was funded by Avego, and started in June 2011. At the time of launch of this phase, many improvements had been made to the system: the pre-screening process was removed, a driving app for windows phone holders was developed (WP7), and riders were able to book rides using SMS. The program was focused on a route connecting Capitol Hill in Seattle to Overlake in Radmond, where the Microsoft campus was located. This route was selected because out of the 1000 people registered in phase 1, 300 of them lived in Capitol Hill, and worked for Microsoft, which made this route the best transport option for them. Although transit and employee-provided transport options were available, but they required long transit times or long distances on foot, making Avego's Real-Time Ridesharing (RTR) an optimal choice.

Many incentives were considered during the length of this phase at different times, ranging from gas cards to gift cards, larger prizes for drawings, free Avego credits, etc., some of which proved to be more effective than the others. In addition, Avego provided a Guaranteed Ride Home (GRH), which included a free shared van service, during the peak hours. Use of the HOV lanes, specially the 3+ ones, was one of the other incentives for the participants. During the 10 week trial of phase 2, 400 trips were logged from 235 users, with the peak repeat user rate of 65%.

For commuters who were not familiar with carpooling, an information session was held where commuters could book their first ride with an Avego employee, and then with a regular Go520 driver. It should be noted than in general, only a small proportion of people have such a high trust threshold, and for places where carpooling is already a practice, the trusting issue is not a big barrier. One of the potential obstacles in this phase was abundance of free commute options for Microsoft staff. For the riders to participate in RTR, the benefits of the program should have proved to surpass the free commute.

Avego learned three main lessons from the Go520 project: 1) involvement of the TDM (Transportation Demand Management) agencies and employers is very effective in building trust, 2) the main incentives for participation in RTR was saving money and time, 3) to build the critical mass required to run TDM, crowded corridors are a good choice.

2.1.10 WeGo, California (Wego rideshare (2013))

This $1.5 million project was conducted in San Francisco Bay area. The project was administrated by the Metropolitan Transportation Commission (MTC), in conjunction with the Sonama County Transportation Authority, the Climate Protection Campaign in Sonama County, the Transportation Authority of Marine, and the Contra Costa Transportation Authority. The technology for this project is provided by Avego, and the requirements to use the system include smartphones for drivers, phones for riders, and the internet. Each program has its own characteristics, and offers a specific reward system.

WeGo Sanoma: The purpose of this project is three-fold: 1- to reduce parking demand and increase the number of shared commute trips. 2- to reduce GHG emissions. 3- to reach the critical mass required to run the system.

All new users receive $20 in Avego credit to use toward receiving rides. Riders pay a $1 booking free plus $0.2 per mile for the first 15 miles and $0.08 per mile thereafter. The micropayments are transferred electronically from the rider to the driver. The driver has the option to provide a free ride.

The WeGo ”More” reward program offers a $5 Starbucks coffee card to the first-time WeGo users. Starting from June of 2013, the program is trying out a new rewarding system. In the month of June participants who use the program for at least 10 days are entered in the draw of 5 $100 gift cards.

The WeGo referral program offers $5 Amazon gift cards to participants who refer other people from the community to the program. The referees will receive the gift card after their referral uses the program for the first time. For those participants who feel charitable, the opportunity has been provided to donate the dollar amount of their rewards to LandPaths.

WeGo Marin: The WeGo Marin offers three types of services: bookings for the RTR minutes before leaving, one-time shared rides booked in advance, or traditional carpools. The RTR service is rolling out for major employers, including Larkspur ferry and Marin college during spring of 2013. As an incentive, participants in the program are entered into a weekly draw of up to $100 in prizes. In addition, participants who engage in the program more than 2 times a week receive a $5 Starbucks coffee card.

Contra Costa: The Contra Costa RTR program offers a guaranteed ride home for the users who go to work as riders. This program also offers weekly and monthly drawing of gift cards.

They also have a public score board that shows the number of drivers, number of rides provided by each driver, and savings in terms of GHG emissions. Up to this date (June 27, 2013), Contra Costa has had 582 shared trips by 59 registered drivers, with the top driver giving 182 rides to a pool of 6 riders. The ridesharing program has led to reduction in CO2 emissions in the amount of more than 14,000 pounds.

2.1.11 Lessons Learned

Figures 2.1 and 2.2 display a summary of the ridesharing systems presented in this section. Although P2P ridesharing systems initially emerged in 1990s, at the time they did not receive enough support from the targeted population to continue operating. The problem with these first attempts was that there was not enough awareness and sufficient marketing regarding the systems and therefore they could not reach the critical mass required to continue running. In addition, the perceived lack of flexibility and convenience, and the safety and security concerns overweighed the incentives such as using HOV lanes.

The lessons learned from these early trials, however, were very useful in developing the more successful ridesharing systems later. Based on these early experiences, the systems emerging later tried to invest more on marketing and building awareness in target groups, consider more tangible incentives such as free parking spots and monetary compensation, provide a guaranteed ride home for the riders who were not able to find a ride back, design fixed payment schemes to avoid awkward negotiations, confirm the matches ahead of time so that people can be certain about their plans, and finally include a pre-screening process for system participants to enhance safety and security.

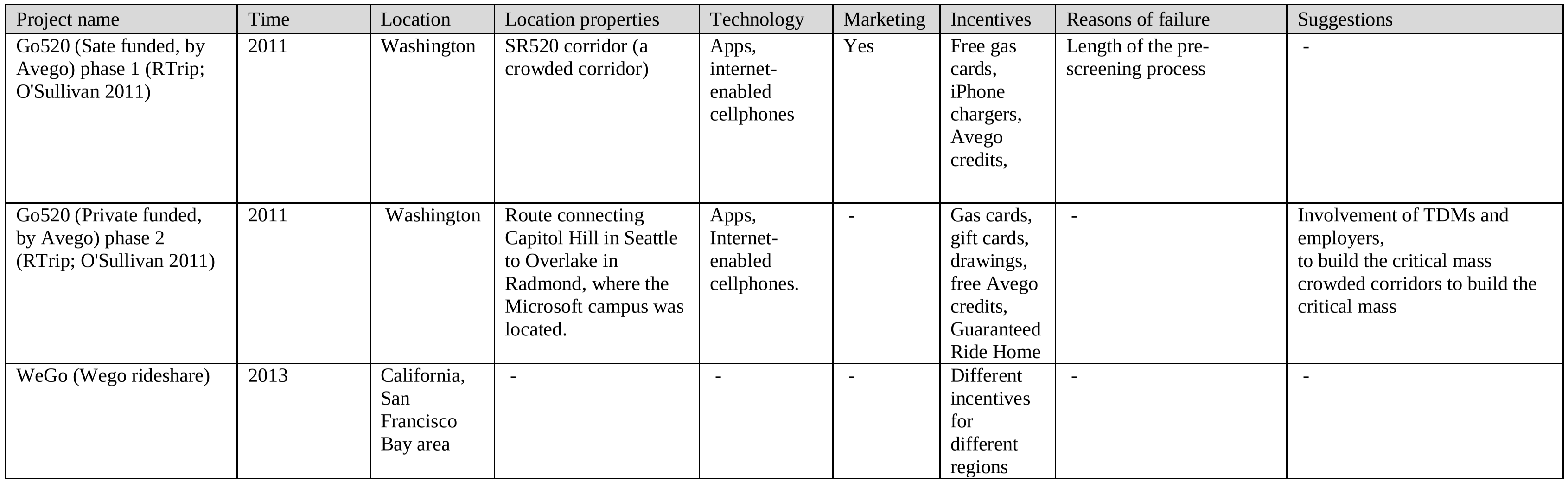
Despite all these new improvements based on the lessens learned from older projects, if it were not for the technological improvements, the more recent ridesharing systems would not enjoy the level of success they are experiencing today. In 1990s, communication between system participants occurred using individuals' private phones or public kiosks implemented in some regions for this purpose. Through these years, using email for establishing communication was considered an extreme technological contribution. The ridesharing systems today mostly run using mobile apps and ridesharing websites, which make using the systems much simpler.

Another very important lesson learnt through years of ridesharing was that having a targeted group of people for a ridesharing system is crucial. Offering such programs in crowded corridors or for certain employers makes it easier to spread the word and advertise the system, and may assure use of sufficient participants to keep the system up and running.

The existence of private ridesharing companies such as Carma (former Avego) and Zimride today, as opposed to the early efforts that were mostly government-funded, implies that peer-to-peer ridesharing is little by little finding its way through society and turning into a profitable business.

Project name	Time	Location	Location properties	Technology	Marketing	Incentives	Reasons of failure	Suggestions
Bellevue Smart Traveler (Haselkorn et al., 1995)	Phase 1: 1993 Phase 2: 1995	Washington	Dense employment location in the city's downtown area, where the majority of commuters travel	Voice-activated matching service	-	-	Lack of flexibility, lack of convenience, Time consuming searching and confirming process, Lack of critical mass, Lack of awareness	Tangible incentives, Guaranteed ride home, Confirming rides one hour before the departure time, Prescreening process, Stations where people can meet
Los Angeles Smart Traveler (Giuliano et al., 1995)	1994	Los Angeles, California	-	Kiosks	-	-	Lack of sense of security, No marketing, Ineffective technology (voice mail and waiting for return calls)	-
Coachella Valley Transaction Network (Levofsky and Greenberg, 2001)	1994	Riverside, California	-	Kiosks	-	-	lack of interest	-
Sacramento Dynamic Ridesharing project (Kowshik et al. 1993)	1994	Sacrament, California	-	Email and cellphone number	-	-	Security concerns, Inadequate marketing, Lack of proper incentives	Pre-screening process, Fixed payment scheme
Seattle Smart Traveler (Dailey et al. 1999)	1996	Seattle, Washington	University of Washington (familiarity with communication techniques and schedule of classes)	-	Yes (toward the end)	-	-	Marketing
RideNow (Nelson Nygaard Consulting Associates and RideNow, Inc, 2006)	2005	Dublin/Pleasanton California	-	Phone	Yes	Free BART tickets, Guaranteed ride home	Cumbersome system, Short notification time frame than the 15 minutes	Sustained marketing, Parking spaces as an incentive where applicable, An easy-to-use system More lead time
Goose Networks Co. (Heinrich, 2010)	-	San Francisco, California	For the employees of the Genentech Co.,	SMS texting		$4 per day commute incentive program	Competitive modes(the well-equipped busses offered by Genentech), Required a high level of customer service that was hard to sustain.	-
Avego (Heinrich, 2010; University College Cork (UCC) official student news)	2011	University Cork, Ireland	From Carrigaline to Cork (highly used, other modes not available)	iPhone app, Internet access	-	-	-	-

Figure 2.1: History of ridesharing (part I)



Project name	Time	Location	Location properties	Technology	Marketing	Incentives	Reasons of failure	Suggestions
Go520 (Sate funded, by Avego) phase 1 (RTrip; O'Sullivan 2011)	2011	Washington	SR520 corridor (a crowded corridor)	Apps, internet-enabled cellphones	Yes	Free gas cards, iPhone chargers, Avego credits,	Length of the pre-screening process	-
Go520 (Private funded, by Avego) phase 2 (RTrip; O'Sullivan 2011)	2011	Washington	Route connecting Capitol Hill in Seattle to Overlake in Radmond, where the Microsoft campus was located.	Apps, Internet-enabled cellphones.	-	Gas cards, gift cards, drawings, free Avego credits, Guaranteed Ride Home	-	Involvement of TDMs and employers, to build the critical mass crowded corridors to build the critical mass
WeGo (Wego rideshare)	2013	California, San Francisco Bay area	-	-	-	Different incentives for different regions	-	-

Figure 2.2: History of ridesharing (part II)

2.2 Ridesharing in Perspective

Congestion in urban transportation networks is one of the common problems faced by many countries around the world. In addition to having a direct impact on travel time and fuel consumption, congestion imposes indirect costs by increasing travel time uncertainty, as well as emission levels which adversely affect human health and ecosystems. Managing congestion by expanding the infrastructure is costly and damaging to the environment. An alternative way is to make more efficient use of the existing transportation infrastructure. Public transportation is a conventional way of using the existing capacity on the roads more efficiently.

Transit systems in urban networks mainly include buses and rail services. They typically carry multiple passengers, and therefore can help reduce vehicle miles traveled (VMT) and ease congestion in urban networks. One drawback of transit services is that they operate on fixed routes and schedules, which limits their coverage of the network, both geographically and temporally. Urban transit systems are typically and necessarily designed to satisfy peak period demands. Due to significant peaking behavior of demand, the system capacity is often drastically under-utilized during off-peak periods, which causes significant cost inefficiencies. The government regulations on fares exacerbate the cost concerns. Thus transit services usually fail to act as financially independent entities, and are in need of subsidies.

Para-transit services were originally introduced to run as supplementary services alongside transit, and as a means to increase the flexibility of public transportation. These services are demand-responsive, and usually serve multiple passengers at a time, based on spatiotemporal proximity of the requests they receive. Routes and schedules are fairly flexible and demand-dependent. Para-transit services include all shuttle-like services that serve customers such as airport travelers, employee or student commuters, or the elderly and disabled. Since the passage of the Americans with Disabilities Act of 1990, however, the term para-transit has been used more commonly to refer to the services provided to persons with disabilities, or to the elderly. In this dissertation, we reserve the term para-transit to refer to such services.

Although para-transit services can be beneficiary to demographics they target, these demographics are fairly limited. In addition, most services that fall under this category are offered by non-profit organizations that are supported by federal funding. Due to the limit on the amount of subsidies, it is not possible to extend these services to the general public. These limitations along with the desire for a more comfortable (and possibly quicker) ride have resulted in a much higher demand for private sector alternatives, such as taxis.

Taxis are a private form of demand-responsive transportation alternative. They provide door-to-door transportation, but at a higher cost that not everyone can afford. Shared taxi/limousine/shuttle services provide an opportunity for customers to share their trips, cutting the cost of the journey, and potentially reducing the total miles traveled in the system (which translates into lower environmental impact). Over the years, different shared-use services have been designed. Flexible route transit systems (Quadrifoglio et al. 2008; Li and Quadrifoglio 2010; Qiu et al. 2014), and High-Coverage Point-to-Point Transit (HCPPT, Cortés and Jayakrishnan, 2002) are a few examples. HCPPT is perhaps the first conceptual system that envisaged the option of private drivers offering rides to be shared, and entirely eliminated fixed route transit, though the somewhat misleading word "transit" was used in the name.

In all alternative modes of transportation above, with the exception of HCPPT, drivers work as system employees. Some of the discussed transportation alternatives are more flexible than others in terms of routing and scheduling, and some have the potential to benefit the environment and customers by allowing them to share (parts of) their trips. In the next set of transportation alternatives we discuss, drivers are not employed by the agencies.

A different family of transportation alternatives which has attracted considerable attention during the recent years is founded based on the principle of shared-use mobility (Shaheen and Chan, 2015). These alternatives try to reduce the cost of flexible transportation by reducing (or completely eliminating) the capital investments and the operating costs.

Informal carpooling is one of the first and common forms of trip sharing, in which already-familiar individuals use the same vehicle for their travel, which typically involves the same origin and destination points. Examples include parents who take turns in taking their children to school, colleagues who travel to work together, etc. In this form of carpooling, vehicles belong to individuals, and the driver of the vehicle has personal interest in the trip, regardless of whether additional passengers are present in the car. The incentives for informal carpooling could be saving time through carpool lanes, or not having to drive one's own vehicle every single day, if participants take turns in driving. This form of carpooling is usually pre-arranged, and happens among individuals who share commonalities beyond the time and location of their trips.

Transportation Network Companies (TNC), such as Uber and Lyft, are among the more recent faces of shared-use mobility alternatives. TNCs use private vehicle owners and their personal vehicles to provide flexible and on-demand transportation services. These individuals work for the company as independent contractors, as opposed to employees. This substantially reduces the cost of capital and human resources, while generating revenue for both the company and the drivers. Essentially, the basic services provided by TNCs act as a lower-cost alternative to taxi services, but not only do they not address the problem of increasing travel demand and congestion, but they add to it. In terms of sustainability, TNCs impose the same cost to the environment as taxi companies do. Recently, two of the more prominent TNCs, Uber and Lyft, have introduced sharing services Uberpool and Lyft Line, respectively. Such services can certainly reduce the cost and environmental impacts of the base services.

Peer-to-peer (P2P) ridesharing aims at capturing the benefits of TNCs while alleviating their adverse impact on the environment. These systems are founded on the principle of sharing economy. Sharing economy, also known as collaborative consumption, is a fairly old concept that focuses on the benefits obtained from sharing resources (products or services) that would otherwise go unused. This economic model has gained more popularity in the recent years, giving birth to many P2P services in different fields (for examples, refer to Böckmann, 2013). The advent of the internet has extended the domains of sharing economy to global populations, and has highlighted its benefits. Moreover, new computer platforms allow easy and quick development of companion mobile applications that facilitate sharing economy.

Similar to TNCs, drivers in a P2P ridesharing system use their own vehicles to carry passengers, and do not work as agency employees. Contrary to the TNC operations, drivers in a P2P ridesharing system are making trips to perform activities of self interest (as in the case of informal carpooling), i.e., they do not roam around the city only to pick up and deliver passengers. This setting can lead to services that are more environmentally-friendly and cost-efficient compared to the sharing services offered by TNCs.

The overwhelming success and good acceptance of TNCs by the public suggest a bright future for more environmentally-friendly ridesharing systems. Since founded in 2009, Uber has managed to expand its operations in 75 countries and more than 450 cities worldwide. According to business insider, in 2014 Uber had more than 160,000 registered driver-partners in the US, and the number of new drivers subscribing to Uber each month increased exponentially, approaching 40,000 (Business Insider, 2015). The company reportedly had \$1.5 billion revenue in 2015 (Carson, 2016).

Table 2.1 summarizes the attributes of the transportation alternatives discussed above. This table suggests that ridesharing systems have the potential to outperform other alternatives in terms of cost, flexibility, and impact on the congestion and the environment, if they succeed to operate as financially independent systems.

Table 2.1: Transportation alternatives differing in flexibility, cost, and environmental impacts

Transportation	alternatives	Transit	Para-Transit (disabled)	Taxi	Shared Taxi/Van	Informal Carpooling	Transportation network companies (TNC)	P2P Ridesharing
Drivers	Employees	✓	✓	✓	✓			
	Peer to riders					✓		✓
Arrangement	Pre-arranged	✓	✓	✓	✓	✓		✓
	On-demand		✓	✓	✓		✓	✓
Multiple passengers		✓	✓		✓	✓	✓	✓
Financially self-sufficient			✓	✓	✓	✓	✓	?
Cost to customers		Low	Low	High	Medium	Low	Low	Low
Flexibility		Low	Medium	High	Medium	Medium	High	High

Table 2.2: Literature on P2P ridesharing

Author	Flexible Paths	Multi-Hop	Multiple Riders	Formulation/ Solution Algorithm	Optimal Solution
Wolfler Calvo et al. (2004)	✓		✓	Greedy heuristic	
Baldacci et al. (2004)	✓		✓	Optimization	✓
Dusan and Mauro (2005)	✓		✓	Bee colony optimization (meta-heuristic)	
Agatz et al. (2009)	✓	✓		Optimization	✓
Agatz et al. (2011)	✓			Optimization	✓
Herbawi and Weber (2011a)		✓		Genetic/Evolutionary Algorithms	
Herbawi and Weber (2012)	✓		✓	Exact formulation + Heuristic solution	
Di Febbraro et al. (2013)	✓		✓	Optimization	✓
Ghoseiri (2013)		✓	✓	Exact Formulation + Heuristic solution	✓
Schaub et al. (2014)			✓	Bipartite Matching	✓

2.3 P2P Ride-Matching

The Multi-hop P2P ride-matching problem can be formulated as a special case of the general pick-up and delivery problem (GPDP). GPDP consists of devising a set of routes to satisfy transportation requests with given loads, and origin/destination locations. Vehicles that operate these routes each have a certain origin, destination, and capacity (Savelsbergh and Sol (1995)).

The dial-a-ride problem (DARP) is a special case of the GPDP, where all vehicles share the same origin and destination depot, and the loads to be transported are people. Although DARP is usually used in systems that aim at transporting elderly or handicapped people, this problem is very close to the ride-matching problem in ridesharing systems.

In its basic form, DARP considers a depot where a fleet of homogeneous vehicles start their trips in the morning, and to which they return at the end of their shifts. Each passenger is assumed to make the entire trip in the same vehicle, i.e. the possibility of transfers between vehicles is not considered. Variants of DARP that are more application-friendly consider time windows for the pick-up and delivery of passengers (Jaw et al., 1986; Psaraftis, 1983). Cordeau and Laporte (2007a,b) provide an overview of the literature on DARP.

In reality, the problem of transporting passengers is often more complex than the basic form of DARP. Some agencies have their fleet located at stations throughout their operating area. This has motivated the development of the multi-depot formulation for DARP (MD-DARP) (Cordeau and Laporte 2007a). Recently, Carnes et al. (2013) and Braekers et al. (2014) have added heterogeneity into the mix, and studied the multi-depot heterogeneous DARP (MD-H-DARP). These studies include heterogeneity among vehicles (multiple depots, capacity, level of service, and operating costs) as well as passengers (level of care, accompanying individuals, required resources, and total number of passengers).

An additional degree of flexibility that has recently been added to the original DARP is the possibility for passengers to transfer between multiple vehicles/modes of transportation, leading to the emergence of the DARP with transfers (DARPT). Masson et al. (2014a), Stein (1978), and Liaw et al. (1996) are the only papers that have studied this variant of the problem, to the best of our knowledge. Masson et al. (2014a) limit the number of potential transfers to one. Stein (1978) does not put a constraint on the capacity of vehicles, and works with demand at an aggregate level, rather than the individual passengers' travel desires. Liaw et al. (1996) use heuristic algorithms to propose multi-modal routes to para-transit users. In their study, they try to route para-transit vehicles to carry passengers from their homes to bus stops, and from bus stops to their destinations.

The P2P ride-matching problem has attracted attention in academia only in the very recent years. Ride-matching problems share some of the characteristics of the more advanced DARPs, such as multiple depots and heterogeneous vehicles and passengers. Drivers in ridesharing systems are traveling to perform activities, and have distinct origin and destination locations (multi-depot), different vehicle capacities (heterogeneity), and rather narrow travel time windows. These factors can lead to the matching problems in ridesharing systems being spatiotemporally sparse, in general. One characteristic that differentiates the ride-matching problem from DARP is the fact that the set of vehicles in a ridesharing system is neither fixed (i.e. not a certain fleet size is available on a regular basis), nor deterministic (i.e. the system does not know in advance the time windows and origins and destinations of drivers' trips). In addition, drivers who make their vehicles available in ridesharing systems are peers to the passengers who are looking for rides, and therefore measures of quality of service that are reserved only for passengers in DARPs should be extended to drivers as well in ridesharing systems.

Agatz et al. (2012) and Furuhata et al. (2013) classify ride-sharing systems based on different criteria, and discuss the challenges ridesharing systems face. In its simplest form, the ride-matching problem matches each driver with a single rider. This can be modeled as a maximum-weight bipartite matching problem that minimizes the total rideshare cost (Agatz et al., 2011).

There are also ride-matching problems that are more complex and try to take advantage of the full unused capacity of vehicles by allowing multiple riders in each vehicle. This form of ridesharing is similar to the carpooling problem where a large employer encourages its employees to share rides to and from work (Baldacci et al., 2004, Teodorović and Dell'Orco, 2005, Schaub et al., 2014, and Wolfler Calvo et al., 2004).

The taxi-sharing problem, as formulated by Hosni et al. (2014), also tries to reduce the cost of taxi services by having people share their rides. Herbawi and Weber (2012) have studied the problem of matching one driver with multiple riders in the context of ridesharing, and have proposed non-exact evolutionary multi-objective algorithms. Di Febbraro et al. (2013) formulate an optimization problem to model the one-to-many ridesharing systems (in which each rider is paired with only one driver, though each driver can carry multiple riders), and use optimization engines to solve it. Stiglic et al. (2015) manage to increase the number of served riders by having riders walk to meeting points, where multiple riders can be picked up by a driver. The number of stops for each driver, however, is limited to a maximum of two.

Herbawi and Weber (2011a,b) model another variant of the ride-matching problem in which a single rider can travel by transferring between multiple drivers. They propose a genetic algorithm to solve this one-to-many matching problem. Masson et al. (2014b) study a similar problem in a multi-modal environment, where goods are carried using a combination of excess bus capacities and city freighters. They propose an adaptive large neighborhood heuristic algorithm to solve the problem. Coltin and Veloso (2014) show the fuel efficiency that including transfers in the riders' itineraries can offer, using three heuristic algorithms.

Many-to-many matching problems allow drivers to have multiple passengers on board at each point in time, and riders to transfer between drivers (Agatz et al., 2009). Cortés et al. (2010) were the first to formally formulate a many-to-many pick-up and delivery problem. They introduced an exact branch-and-cut solution method. The largest example they solved, however, consisted of 6 requests, two vehicles, and one transfer point. To the best of our knowledge, there are only two studies that model many-to-many ridesharing systems. Agatz et al. (2009) is one of the first to take an optimization approach toward modeling many-to-many ridesharing systems. In their study, the authors discuss modeling multi-modal ridesharing systems that allow for transfers between different modes of transport. However, they do not discuss a solution methodology. Ghoseiri (2013) formulates a mixed integer problem (MIP) to model the many-to-many ridesharing system. Their proposed solution heuristic limits the number of transfers to a maximum of two.

In addition to multi-hop ridesharing, a solution methodology that can handle unlimited number of transfers can be used to optimize multi-modal transportation networks, where, for example, ridesharing is combined with public transportation. In such a scenario, each public transport line acts as a driver with a fixed route in the ridesharing system. In case of a multi-modal network, allowing for higher numbers of transfers becomes a necessity, and an algorithm that can accomplish routing and scheduling of passengers in real-time becomes necessary.

Chapter 3

Peer-to-Peer Ridesharing System

The ridesharing system defined in this book contains a set of participants P. These participants are divided into a set of riders, R, who are looking for rides, and a set of drivers, D, who are willing to provide rides ($P = R \cup D$). Drivers may have different incentives to participate in the ridesharing system, including monetary compensation, using high occupancy vehicle (HOV) lanes, or reduced-cost parking, among others. Different driver incentives for participating in ridesharing can translate into different objective functions for the corresponding ride-matching problem.

The system finds matches for riders by optimally routing drivers in the network. In order to guarantee a high quality of service, both riders and drivers provide a time window to specify the start and end of their trips, and a maximum ride time. Furthermore, riders can specify the maximum number of transfers they are willing to make, and drivers can put a limit on the number of riders they want to have on board at each moment in time. The term "real-time" emphasizes the capability of the system to make ride-matches in a short period of time, for implementation with frequent re-optimizations using newer data over time.

To facilitate pick-ups and drop-offs, a set of stations, S, are identified in the network. Stations are pre-specified locations where participants can start and end their trips, and riders can switch between drivers and/or to other modes of transport, such as transit. Strategic identification of stations is central to the performance of the system. Lessons learned from the previous P2P ridesharing systems suggest that it is better for riders to be picked up/dropped off at pre-specified locations, rather than their homes (or the exact location where their trips start/end) for two reasons (Heinrich, 2010). First, these locations could be hard to find for drivers, and therefore riders could miss their scheduled rides. In addition, drivers could have a hard time finding an appropriate location to park their vehicles. Second, some drivers and riders would understandably be reluctant to reveal their home address to others. In addition, as shown by Stiglic et al. (2015), introducing stations can increase the number of successful matches. Note, however, that these stations could be fairly small in its infrastructural scope, in that they could be spots where a vehicle or two could stop and a few riders could wait.

Every active participant p of the rideshare system provides to the system their origin and destination stations (OS_p and DS_p respectively), the earliest acceptable time to depart from the origin station, T_p^{ED}, the latest acceptable time to arrive at the destination station, T_p^{LA}, and their maximum ride time, T_p^{TB}. The travel time window for participant p is defined as $TW_p = [T_p^{ED}, T_p^{LA}]$. In addition, each participant is asked to provide a notification deadline by which they need to be informed of any matches made for them. Some of these parameters are displayed in Figure 3.1.

Drivers should announce the capacity of their vehicles, C_d. This could simply be the physical capacity of the vehicle, or the maximum number of riders the driver is willing to carry in their vehicle at any point in time. Riders can specify the maximum number of transfers (change of vehicles), V_r, that they are willing to make to get to their destinations.

The goal of the ridesharing system in this dissertation is to maximize the matching rate (although we discuss various objective functions in Chapter 4). For now, let us assume that

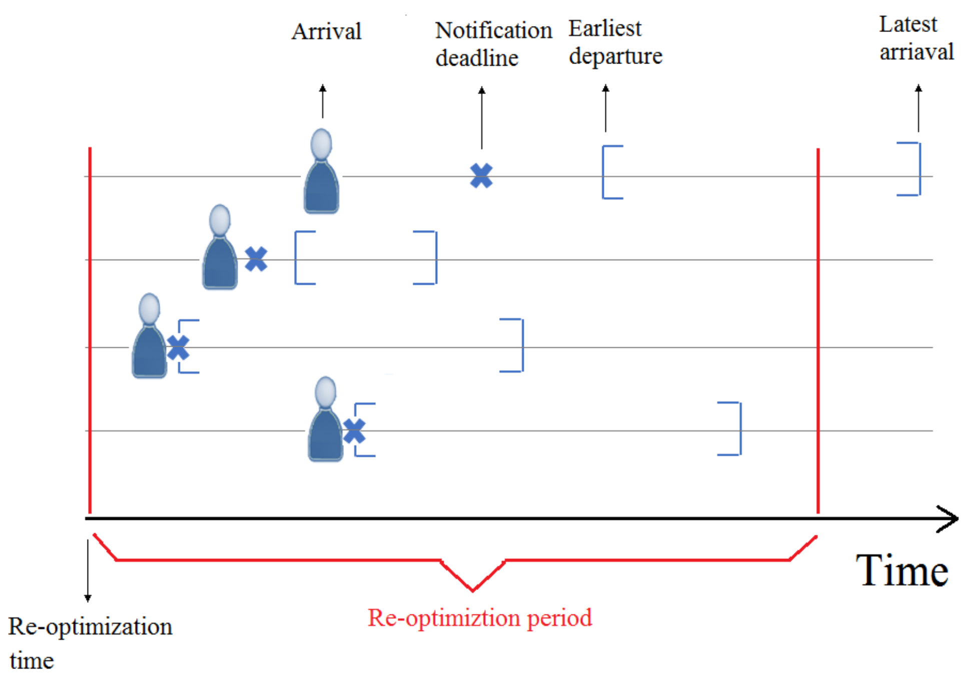


Figure 3.1: Inputs from Participants

drivers leave the route choice to the system. This does not preclude the case of drivers who want to follow their own fixed routes, as those routes can be specified as successive nodes and entered into the formulation as fixed parameters.

3.1 Implementation Strategies

There are multiple implementation strategies that the operator of a ridesharing system can consider. The implementation strategy for a system depends greatly on the nature of the requests received by the system, specifically on the lag between the time participants register in the system, their notification deadlines, and their earliest departure times.

A system whose participants' registration times are very close to their notification deadlines, is highly dynamic. Services offered by ridesourcing companies such as Uber and Lyft are of this nature. For such systems, a very fast matching algorithm is necessary. In chapter

6, we introduce a Dynamic Programming (DP) algorithm that is capable of handling the matching in such highly dynamic systems by serving riders on a First-Come First-Served (FCFS) basis.

Another implementation strategy is to serve ride requests on a rolling time-horizon basis (Figure 3.1). In such an implementation, the system operator optimizes the system periodically, at pre-specified points in time called "re-optimization times" (say every 30 minutes starting from 12:00 A.M.). At each re-optimization time, a matching problem is solved that includes all the riders whose travel time windows intersect with the period for which the system is being optimized. After solving the optimization problem, the itineraries of participants whose notification deadlines lie within the optimization period is fixed, and announced to them. In Chapter 7, we propose a decomposition algorithm that can optimally solve this many-to-many matching problem.

In Chapter 7, we also discuss the operations of a system that falls in between the two implementation strategies discussed above, where a subset of participants register their trips ahead of their actual departure times, and the rest of participants register their trips not long before they plan to leave their origin stations. In such a scenario, the decomposition algorithm in Chapter 7 can be used for individuals who announce their trips ahead of time, and the DP algorithm in Chapter 6 can be used for those who arrive at the system in real-time.

In a highly dynamic system where a participant's registration time and notification deadline are close, it could be the case that the participant's earliest departure time is further ahead in time, or their maximum ride time is higher than their shortest path travel time. For such instances, we propose a P2P ride exchange mechanism in Chapter 8, which enables matched riders to sell their current itineraries to real-time riders in exchange for a less convenient itinerary, and a monetary compensation. The P2P exchange mechanism can also be used in a mixed system, where a subset of participants announce their trips ahead of time, and a subset arrive in real-time. In such a system, the P2P ride exchange mechanism gives the real-time riders a higher chance of being served.

Another way to increase the matching rate in a ridesharing system is to use historical data to forecast the arrival of participants who tend to register their trips in real-time, and route drivers to put them in spatiotemporal proximity of their trips. Chapter 9 is dedicated to this concept. In this chapter, we introduce the stochastic ride-matching problem and propose a solution methodology to solve this problem efficiently.

Chapter 4

Mathematical Formulation

The role of the ride-matching problem is to devise itineraries that can take riders to their destinations by optimally routing drivers. Itineraries have to comply with the riders' specified maximum number of transfers, and the capacity of drivers' vehicles, as well as the participants' travel time windows and maximum ride times. The ride-matching problem will devise itineraries for the matched riders in the system, and all drivers, matched or not.

4.1 Time-Expanded Network

We formulate the general many-to-many ride-matching problem in a time-expanded network. By introducing stations, we discretize the two-dimensional network into a single space dimension. In addition, we discretize the study time horizon into an ordered set of indexed time intervals of small duration, Δt, to allow for using time-dependent travel-time matrices. We define set T_p to contain the indices for all time intervals within the range $TW_p = [T_p^{ED}, T_p^{LA}]$ for participant p.

In a network discretized in both time and space, we define node i, n_i, as a tuple (t_i, s_i), where

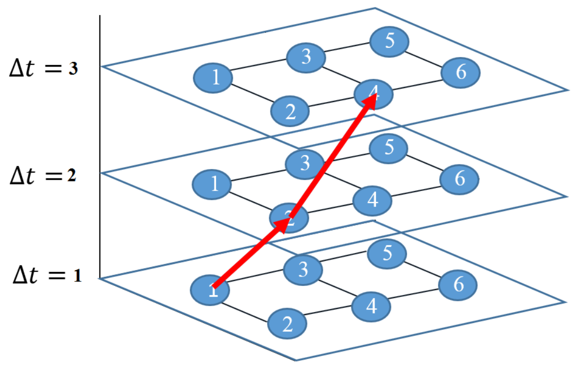


Figure 4.1: Example of links in a time-expanded network

t_i is the time interval one may be located at station s_i. Subsequently, a link ℓ is defined as $(n_i, n_j) = (t_i, s_i, t_j, s_j)$, where t_i is the time interval one has to leave station s_i, in order to arrive at station s_j during time interval t_j. We generate links between neighboring stations only, i.e. a link exists between stations s_i and s_j if traveling from s_i to s_j does not require passing through another station. Naturally, the links are determined by the travel time matrix, which could be time-dependent, and travel times used must ensure that a traveler leaving at the very end of the time interval of t_i can arrive at least by the very end of the time interval t_j. We denote the set of links by L.

Figure 4.1 demonstrates an example of a link in a time-expanded network. If a participant leaves station 1 at time interval 1, they arrive at station 2 at time interval 2. The starting and ending nodes of this trip are $n_s = (t_s, s_s) = (1, 1)$, and $n_e = (t_e, s_e) = (2, 2)$ respectively. The resulting link is $\ell = (t_s, s_s, t_e, s_e) = (1, 1, 2, 2)$. In this figure, we can also see the link $(2, 2, 3, 4)$. These two links together form a path that leaves station 1 at time interval 1, and arrives at station 4 at time interval 3.

4.2 Main Formulation

To mathematically model the ride-matching problem, we use four sets of decision variables, as defined in equations (10.1)-(4.4).

$$x_\ell^d = \begin{cases} 1 & \text{Driver } d \text{ travels on link } \ell \\ 0 & \text{Otherwise} \end{cases} \tag{4.1}$$

$$y_\ell^{rd} = \begin{cases} 1 & \text{Rider } r \text{ travels on link } \ell \text{ with driver } d \\ 0 & \text{Otherwise} \end{cases} \tag{4.2}$$

$$z_r = \begin{cases} 1 & \text{Rider } r \text{ is matched} \\ 0 & \text{Otherwise} \end{cases} \tag{4.3}$$

$$u_r^d = \begin{cases} 1 & \text{Driver } d \text{ contributes to the itinerary for rider } r \\ 0 & \text{Otherwise} \end{cases} \tag{4.4}$$

Equation (10.3a) presents the objective function of the problem. A ride-matching problem can have various objectives, ranging from maximizing profits to minimizing the total miles/hours traveled in the network. This objective can vary depending on the type of the agency that is managing the system (public or private), the level of acceptance of the system in the target community, and the ridesharing incentives. For a ridesharing system in its infancy, it seems logical to maximize the matching rate. We use this objective for the ridesharing system in this dissertation.

The first term in the objective function (10.3a) maximizes the total number of served riders, while the second term minimizes the total number of transfers in the system, and is added only for the purposes described in Proposition A (in the Appendix A). The weight W_r should be set to a proper value (any value smaller than $\frac{1}{V_r}$, where V_r is the maximum number of transfers rider r is willing to make) to ensure that serving the maximum number of riders remains the primary objective of the system.

The sets of constraints that define the ridesharing system are presented in (10.3b)-(4.5m). Constraint sets (10.3b)-(10.3d) route drivers in the network. Constraint set (10.3b) directs drivers in set D out of their origin stations, and (10.3c) ensures that they end their trips at their destination stations. Note that we do not use a separate set of constraints to enforce the travel time windows for drivers. Such constraints are satisfied automatically, since we limit the set of links in constraint sets (10.3b) and (10.3c) to the ones whose time intervals are within the driver's travel time window (in set T_d).

Constraint set (10.3d) is for flow conservation, enforcing that a driver entering a station in a time interval, exits the station in the same time interval. Notice that participants might not physically leave a station. Members of the link set L in the form $(t, s, t+1, s)$ represent the case where a participant is physically remaining at station s for one time interval, but technically leaving node (t, s) for node $(t+1, s)$. In addition, constraint set (10.3d) enforces the multi-hop property of the ridesharing system. Riders can enter a node with a driver, and exit it with a different driver, suggesting a transfer between the two drivers (or modes of transportation). Constraint set (4.5e) limits the total travel times by drivers based on their maximum ride times.

$$\text{Max} \qquad \sum_{r\in R} z_r - \sum_{r\in R} W_r \sum_{d\in D} u_r^d \tag{4.5a}$$

$$\sum_{\substack{\ell\in L:\\ s_i=OS_d;t_i,t_j\in T_d}} x_\ell^d \quad - \sum_{\substack{\ell\in L:\\ s_j=OS_d;t_i,t_j\in T_d}} x_\ell^d = 1;\ \forall d\in D \tag{4.5b}$$

$$\sum_{\substack{\ell\in L:\\ s_j=DS_d;t_i,t_j\in T_d}} x_\ell^d \quad - \sum_{\substack{\ell\in L:\\ s_i=DS_d;t_i,t_j\in T_d}} x_\ell^d = 1;\ \forall d\in D \tag{4.5c}$$

$$\sum_{\substack{t_i,s_i\\ \ell=(t_i,s_i,t,s)\in L}} x_\ell^d \quad = \sum_{\substack{t_j,s_j\\ \ell=(t,s,t_j,s_j)\in L}} x_\ell^d;\ \forall d\in D,\forall t\in T_d,\forall s\in S\backslash\{OS_d\cup DS_d\} \tag{4.5d}$$

$$\sum_{\ell\in L}(t_j-t_i)x_\ell^d \le \frac{T_d^{TB}}{\Delta t};\ \forall d\in D \tag{4.5e}$$

$$\sum_{d\in D'}\sum_{\substack{\ell\in L:\\ s_i=OS_r;t_i,t_j\in T_r}} y_\ell^{rd} - \sum_{d\in D'}\sum_{\substack{\ell\in L:\\ s_j=OS_r;t_i,t_j\in T_r}} y_\ell^{rd} = z_r;\ \forall r\in R \tag{4.5f}$$

$$\sum_{d\in D'}\sum_{\substack{\ell\in L:\\ s_j=DS_r;t_i,t_j\in T_r}} y_\ell^{rd} - \sum_{d\in D'}\sum_{\substack{\ell\in L:\\ s_i=DS_r;t_i,t_j\in T_r}} y_\ell^{rd} = z_r;\ \forall r\in R \tag{4.5g}$$

$$\sum_{d\in D'}\sum_{\substack{t_i,s_i:\\ \ell=(t_i,s_i,t,s)\in L}} y_\ell^{rd} = \sum_{d\in D'}\sum_{\substack{t_j,s_j:\\ \ell=(t,s,t_j,s_j)\in L}} y_\ell^{rd};\ \forall r\in R,\forall t\in T_r,\forall s\in S\backslash\{OS_r\cup DS_r\} \tag{4.5h}$$

$$\sum_{d\in D'}\sum_{\ell\in L}(t_j-t_i)y_\ell^{rd} \le \frac{T_r^{TB}}{\Delta t}\forall r\in R \tag{4.5i}$$

$$\sum_{r\in R} y_\ell^{rd} \le C_d x_\ell^d;\ \forall d\in D'\forall \ell\in L \tag{4.5j}$$

$$u_r^d \ge y_\ell^{rd};\ \forall r\in R,\forall d\in D,\forall \ell\in L \tag{4.5k}$$

$$u_r^d \le \sum_{\ell\in L} y_\ell^{rd};\ \forall r\in R,\forall d\in D \tag{4.5l}$$

$$\sum_{d\in D} u_r^d - 1 \le V_r;\ \forall r\in R \tag{4.5m}$$

Rider r's itinerary is determined by variable y_ℓ^{rd}. A value of 1 for this variable indicates that the rider is traveling on link ℓ in driver d's vehicle. By definition, this variable implies that a rider should always be accompanied by a driver. However, in reality, a rider does not need to be accompanied when he/she is traveling on a link in the form $(t, s, t+1, s)$, i.e. staying at station s, waiting to make a transfer. To incorporate this element into the formulation, we introduce the dummy driver d'. The dummy driver does not have a real origin or destination in the network. The set of links used by the dummy driver is also different from the members of the link set L. We define the set of links for the dummy driver as $L' = \{(t, s, t+1, s), \forall (t, s) \in T \times S\}$. This set includes all the links that represent staying at a station for one time interval. We define set D' to include the dummy driver in addition to the set of drivers, D, i.e. $D' = \{D \cup d'\}$.

Constraint sets (10.3e)-(10.3g) route riders in the network, and are analogous to (10.3b)-(10.3d), except for a small variation. While drivers, matched or not, will receive an itinerary, this is not the case for the riders. Only the riders who are successfully matched will receive itineraries. This difference is reflected in the formulation by replacing 1 on the right hand side of constraint sets (10.3b)-(10.3c) by z_r in constraint sets (10.3e)-(10.3f). Constraint set (4.5i) sets a limit on the riders' maximum ride times.

Constraint set (4.5j) serves two purposes. First, it ensures that riders are accompanied by drivers throughout their trip. Second, it ensures that vehicle capacities are not exceeded. Constraint sets (4.5k)-(4.5m) collectively set a limit on the total number of transfers for each rider. Constraint sets (4.5k) and (4.5l) register drivers who contribute to each rider's itinerary (refer to Proposition B in the Appendix B). Constraint set (4.5m) restricts the number of transfers by each rider (refer to Proposition A in the Appendix A). Finally, all decision variables of the problem defined in (10.1)-(4.4) are binary.

4.3 Problem Variants

It is possible to expand the formulation in model (4.5) to include various other objectives and/or system characteristics. In this section, we review how some minimal changes can make the formulation suitable for different circumstances.

4.3.1 Expanding the Objective Function

Although the formulation in model (4.5) is defined for a ridesharing system in its infancy, it is easy to include additional terms in the objective function or introduce additional decision variables and constraint sets to cover different objectives ridesharing systems may have throughout their lifetime. For instance, we can minimize the total travel time by riders and drivers by adding terms $\sum_{d\in D}\sum_{r\in R}\sum_{\ell\in L}(t_j - t_i)y_\ell^{rd}$ and $\sum_{d\in D}\sum_{\ell\in L_d}(t_j - t_i)x_\ell^d$ to the objective function, respectively. We can also further penalize waiting times (during transfers) for riders by adding the term $W\sum_{d\in D}\sum_{r\in R}\sum_{\ell=(t_i,s_i,t_j,s_j)\in L:s_i=s_j} y_\ell^{rd}$ to the maximization objective function, where W is a negative penalty for each interval of waiting.

4.3.2 Targeting Drivers

It is possible to minimize/maximize the number of matched drivers by introducing the decision variable z'_d that takes the value of 1 is driver d is matched, and 0 otherwise. In this case, the unit values on the right hand sides of constraint sets (10.3b) and (10.3c) should be replaced with z'_d, and the term $W'\sum_{d\in D} z'_d$ should be added to the objective function.

4.3.3 Including Service Times

In the formulation presented in model (4.5), we do not consider an exclusive service time for participants. This is a realistic assumption, since we are concerned with people and not goods. However, if service times are required due to practical considerations, they can be easily integrated by making small changes in the definition of the link sets and constraint sets (10.3d) and (10.3g). Originally, we defined a link between two stations s_i and s_j if the two stations were neighboring stations in the network. To account for service times, we have to redefine the link sets to include links between any two stations. If a rider travels from their origin station to station k, and then from station k to their destination station, this implies a transfer at station k. To include service times, equations (10.3d) and (10.3g) should be replaced with constraint sets (4.6) and (4.7) respectively, where t_s is the service time (in number of time intervals).

$$\sum_{\substack{t_i,s_i \\ \ell=(t_i,s_i,t,s)\in L}} x^d_\ell = \sum_{\substack{t_j,s_j \\ \ell=(t+t_s,s,t_j,s_j)\in L}} x^d_l; \ \forall d \in D, \forall t \in T_d, \forall s \in S\backslash\{OS_d \cup DS_d\} \tag{4.6}$$

$$\sum_{d\in D'} \sum_{\substack{t_i,s_i: \\ \ell=(t_i,s_i,t,s)\in L}} y^{rd}_\ell = \sum_{d\in D'} \sum_{\substack{t_j,s_j: \\ \ell=(t+t_s,s,t_j,s_j)\in L}} y^{rd}_\ell; \ \forall r \in R, \forall t \in T_r, \forall s \in S\backslash\{OS_r \cup DS_r\} \tag{4.7}$$

4.3.4 Individual Rationality Constraints

The individual rationality constraints ensure that all the parties involved would benefit from participating in the system. Adding constraint sets (4.8a)-(4.8c) to model (4.5) ensures that riders and drivers would benefit from participating in the system (given a distance-based fare charged to riders, and transfered to drivers), and that the total VMT would not increase as a result of the operations of the ridesharing system.

$$\sum_{r\in R}\sum_{\ell\in L}(\theta_m\zeta_\ell^m y_\ell^{rd} + \alpha\theta_t\zeta_\ell^t y_\ell^{rd}) \geq \sum_{\ell\in L}\zeta_\ell^m\, x_\ell^d - SP_d^m + \alpha\big(\sum_{\ell\in L}\zeta_\ell^t\, x_\ell^d - SP_d^t\big) \qquad \forall d\in D \tag{4.8a}$$

$$\sum_{d\in D}\sum_{\ell\in L}\left(\theta_m\,\zeta_\ell^m y_\ell^{rd} + \alpha\theta_t\zeta_\ell^t y_\ell^{rd}\right) + w_r\sum_{d\in D}u_r^d \leq G_r \qquad \forall r\in R \tag{4.8b}$$

$$\sum_{d\in D}\sum_{\ell\in L}\zeta_\ell^m x_\ell^d \leq \sum_{r\in R}SP_r\, z_r + \sum_{d\in D}SP_d \tag{4.8c}$$

Constraint set (4.8a) is the drivers' individual rationality constraint. Here, ζ_ℓ^m and ζ_ℓ^t are the length of link ℓ in miles and hours, respectively. Normalizing the unit distance and time costs of travel for drivers to 1 and α respectively, we assume θ_m and θ_t (both between 0 and 1) to be the proportion of the distance- and time-based costs for which the driver will be compensated, and SP_d^m and SP_d^t to be the lengths of the shortest distance and time paths for driver d, respectively. The left hand side of constraint (4.8a) represents the revenue earned by driver d, and the right hand side of this constraint is the total cost of the detour undertaken by the driver in order to transport passengers. This constraint set ensures that drivers earn a non-negative revenue by participating in the ridesharing system.

Constraint set (4.8b) is the riders' individual rationality constraint. On the left hand side of the constraint, we have the generalized cost of participating in the ridesharing system realized by rider r. This cost is composed of three components: a distance-based cost, a time-based cost, and a penalty associated with discomfort of transfers, ω_r, where all three components are monetized. In case a rider travels with a single driver, ω_r represents the discomfort associated with using the ridesharing system and its inherent uncertainties on finding a match. Higher number of transfers will be penalized linearly, for simplicity, though an argument can be made that this could be non-linear. The right hand side of this constraint is the monetized cost of the rider's most favorable transportation alternative outside of the ridesharing system, in terms of generalized cost. This constraint ensures that the ultimate utility of using the ridesharing system for a given rider is higher than the utility of the rider's all other transportation alternatives, assuming a rider's utility to be the negative of his/her generalized cost.

Constraint set (4.8c) ensures that the total VMT in the network is less than the cumulative cost of individuals traveling using their alternative transportation options if the ridesharing system was not available. Without loss of generality, here we assume this alternative to be for individuals to travel using their personal vehicles through their shortest paths.

Chapter 5

The Ellipsoid Spatiotemporal Accessibility Method (ESTAM)

The goal of the Ellipsoid spatiotemporal accessibility method (ESTAM) is to limit the number of accessible links for each participant, and identify and eliminate drivers who cannot be part of a rider's itinerary due to the lack of spatiotemporal compatibility between their trips. At the outset, it should be noted that this procedure does not limit the search space of the optimization problem in Chapter 4, but only narrows it by cutting down practically infeasible ranges, and therefore it does not affect the optimality of the solution.

As explained before in 4, we present a link, ℓ, as a 4-tuple (t_i, s_i, t_j, s_j). Participants can potentially reach any station in the network in any time interval within their travel time window, making the size of the set of links, L, as large as $O(|T||S|)$, where $|T|$ is the number of time intervals in the study time horizon, and $|S|$ is the number of stations in the network.

The premise of the pre-processing procedure is that the spatiotemporal constraints enforced by maximum ride times and travel time windows of participants limit their access to members of the link set L. We use this information to construct the set of links accessible to riders and drivers, denoted by L_r and L_d respectively.

ESTAM first uses a simple geometric tool to identify the set of stations spatially reachable by each participant. Such stations form the "reduced networks" of participants. Next, ESTAM finds the time intervals during which each station can be reached, forming the "time-expanded reduced networks" (i.e. the set of feasible links) for participants. Finally, ESTAM uses this information to generate the "time-expanded feasible networks" for riders using the set of the riders' feasible links that are reachable by drivers. The three steps of ESTAM are discussed in more detail in the following.

5.1 Generating Reduced Networks

The origin and destination stations, maximum ride times, and travel time windows of participants can be used to define a region in the network in the form of an ellipse, inside which participants have a higher degree of space proximity, i.e. the percentage of accessible stations within this region is at least as high as the same percentage within the entire network. We call the region inside and on the circumference of the ellipse associated with participant p the reduced network of the participant, denoted by G_p (reduced network of rider r/driver d is denoted as G_r/G_d).

The focal points of the ellipse are the participant's origin and destination stations. The length of the major axes between the focal points is the straight distance between the origin and destination stations, and the transverse diameter of the ellipse is an upper-bound on the distance that can be traveled by the participant in $\frac{T_p^{TB}}{\Delta t}$ number of time intervals, within the participant's time window.

We know that for each point on the circumference of an ellipse, sum of the distances from the two focal points is always constant, and equal to the transverse diameter of the ellipse.

By setting the length of this transverse diameter to the maximum distance a participant can travel given their maximum ride time, we ensure that none of the stations outside of the participant's reduced network are accessible to them.

5.2 Generating Time-Expanded Reduced Networks

In the previous step, we formed the reduced networks for the participants. However, not all stations in the participants' reduced networks are necessarily reachable by them, since we have yet to study time proximity of stations. To study time-proximity, we have to find the set of time intervals during which station $i \in G_p$ can be reached by participant p. After the reduced networks are generated, we use the link generation procedure presented in Algorithm 1 to identify the time-expanded reduced networks for participants, and construct sets L_r and L_d.

The algorithm finds the set of links in two steps: a forward movement followed by a backward movement. Both forward and backward movements are iterative procedures. In each iteration, the forward movement generates a set of links originating from one of the stations in the reduced graph. Let us define the set of active stations as S_{act}, and initialize it by the origin station. The algorithm starts with the first member of the active stations set, s_1, which initially is the origin station of the participant (i.e. $s_1 = OS_p$). The participant can leave this station in time intervals within the window $[T_p^{ED}, T_p^{LA} - tt_{(OS_p,DS_p)}]$, where $tt_{(OS_p,DS_p)}$ is the shortest path travel time between participant p's origin and destination stations. $tt_{(OS_p,DS_p)}$ is calculated based on a static travel time matrix. To ensure that no feasible links are eliminated, this static travel time matrix should contain underestimated link travel times, say for example, the travel times during non-peak hours.

After identifying the set of feasible time intervals for the origin station, the origin station is

Algorithm 1 Link reduction procedure

```
**Generate a link set $L_p$ for participant $p$**
**Initialize**
$\forall s \in G_p,\ L(s) = \emptyset$
**Step1. Forward movement**
$S_{act} = \{OS_p\}$
$\bar{S}_{act} = \emptyset$
$T_{OS_p} = \{\frac{T_p^{ED}}{\Delta t} : \Delta t : \frac{T_p^{LA} - tt_{(OS_p, DS_p)}}{\Delta t}\}$
While $S_{act} \neq \emptyset$
    $s_1 \leftarrow S_{act}(1)$
    For $s \in \{S \backslash OS_p\}$
        Set $T_{s_1} = T_{s_1} \cup \{t_1\}$ such that $\exists \ell = (t, s, t_1, s_1) \in L(s),\ \forall t, t_1 \in T_p$
    End For
    $\bar{S}_{act} = \bar{S}_{act} \cup \{s_1\}$
    $S_{act} = S_{act} \backslash \{s_1\}$
    For $s_2 \in S : (s_1, s_2) \in G_p$
        If $s_2 \cap \bar{S}_{act} = \emptyset$
            $S_{act} = S_{act} \cup \{s_2\}$
            For $t \in T_{s_1}$
                $L(s_2) = \{L(s_2) \cup (t, s_1, t + T_{dynamic}(t, s_1, s_2)^{\dagger}, s_2)\}$
            End For
        Else
            For $t \in T_{s_1}$
                If $T_{s_2} \cap \{t + T_{dynamic}(t, s_1, s_2)\} \neq \emptyset$
                    $L(s_2) = \{L(s_2) \cup (t, s_1, t + T_{dynamic^{\dagger}}(t, s_1, s_2), s_2)\}$
                End If
            End For
        End If
    End For
End While
**Step 2. Backward movement**
$L_{del} = \{(t_1, s_1, t_2, s_2) \in L_p : s_2 = DS_p \wedge t_2 > \frac{T_p^{LA}}{\Delta t}\}$
While $L_{del} \neq \emptyset$
    $\ell(t_1, s_1, t_2, s_2) \leftarrow L_{del}(1)$
    $L_{del} = L_{del} \backslash \{\ell\}$
    $L_p(s_2) = L_p(s_2) \backslash \{\ell\}$
    For $s_1 \in G_p$
        For $(t, s) : (t, s, t_1, s_1) \in L_p(s_1)$:
            $L_{del} = L_{del} \cup \{(t, s, t_1, s_1)\}$
        End For
    End For
End While
```

$^{\dagger}T_{dynamic}(t, s_1, s_2)$: travel time between stations s_1 and s_2 at time interval t

eliminated from the set of active stations, outgoing links from the origin station whose end stations are inside the reduced network are identified, and their end stations are added to the set of active stations. At this point, we have information on the starting station (s_1), ending stations ($s_2 : (s_1, s_2) \in G_p$) and the starting time intervals at s_1. The ending time intervals for each s_2 can be easily looked up from a dynamic travel time matrix, completing the information required to construct the set of links originating at s_1. The algorithm proceeds to select the next station from the set of active stations, and generates the set of links originating from this station following the same procedure. We iterate the procedure until the set of active stations in empty.

Figure 5.1 demonstrates an example of the forward movement for a participant who is traveling from station 14 to station 8, with $TW_p = [1, 40](\Delta t = 1 \text{ min})$, maximum ride time of 40 minutes, and shortest path travel time of 38 minutes. Link travel times (in minutes) are shown on the graph. It is assumed that the travel time remains constant on each link. Note that this assumption is made only for simplicity, and using time-dependent travel times would be just as straightforward. The set of links for each station is computed during the forward movement, and is presented in the Figure 5.1. These links are not final, however, and have to be refined during the backward movement.

The backward movement simply scans through the set of links generated for each station by the forward movement and refines these sets by removing the time intervals that are identified as infeasible based on the participant's latest arrival time. In the example shown in Figure 5.1, since the latest arrival time is at $\Delta t = 40$, links with ending time intervals 41 and 42 should be removed from the set of links for the destination station. Tracking back the stations from destination to origin, the time intervals for the stations that have led to the infeasible time intervals at the destination station are identified and removed (see Algorithm 1 for details). For the example in Figure 5.1, after completing the backward movement, the set of links, L_p, is finalized and listed in the figure.

At the end of this step, the set of links for each participant $p \in P$ is identified. These links form the time-expanded reduced network for the participant.

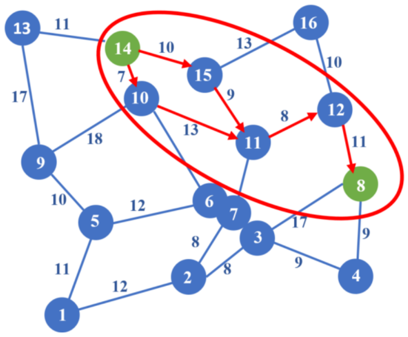
Once the set of links for participants are generated, we can use these sets to reduce the size of some other sets in the optimization problem, namely the set of riders and drivers. We can reduce the size of the rider set R by filtering riders out of the problem based on their accessibility to potential drivers. For a rider to be served, they should have spatiotemporal proximity with at least one driver at both their origin and destination stations. Members of set R who do not enjoy this spatiotemporal proximity can be filtered out.

In addition, in order for a driver d to be able to contribute to the itinerary of a rider r, the intersection of their link sets should not be empty, i.e. $L_r \bigcap L_d \neq \emptyset$. We denote by M the set of tuples $(r, d) \in R \times D$ for whom $L_r \bigcap L_d \neq \emptyset$ and therefore could potentially be matched. For members of set M, we construct a set $L_{rd} = \{L_r \bigcap L_d\}$.

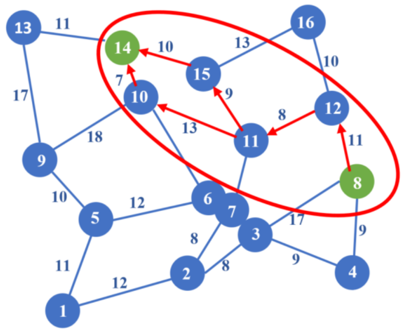
After performing this step, we can replace model (4.5) with a refined version of the optimization problem in model (D.1), presented in Appendix D. This refined model is very similar to model (4.5), with two major differences, thanks to the first two steps of ESTAM: (i) Constraint sets (4.5e) and (4.5i) are now redundant, since the requirement to not exceed the maximum ride times is met, and (ii) input sets in model (D.1) are more refined.

5.3 Generating Riders' Time-Expanded Feasible Networks

In this step, we form the time-expanded feasible network for a rider $r \in R$, by finding sections of the rider's time-expanded reduced network that intersect with at least one driver's time-expanded reduced network. In other words, a rider's time-expanded feasible network consists



Station	14	15	10	11	11	12	8
Pred.	-	14	14	15	10	11	12
L_p	-	(1,14,11,15) (2,14,12,15) (3,14,13,15)	(1,14,8,10) (2,14,9,10) (3,14,10,10)	(11,15,20,11) (12,15,21,11) (13,15,22,11)	(8,10,21,11) (9,10,22,11) (10,10,23,11)	(20,11,28,12) (21,11,29,12) (22,11,30,12) (23,11,31,12)	(28,12,39,8) (29,12,40,8) (30,12,41,8) (31,12,42,8)



Station	8	12	11	11	10	15	14
Pred.	12	11	10	15	14	14	-
L_p	(28,12,39,8) (29,12,40,8) ~~(30,12,41,8)~~ ~~(31,12,42,8)~~	(20,11,28,12) (21,11,29,12) ~~(22,11,30,12)~~ ~~(23,11,31,12)~~	(8,10,21,11) ~~(9,10,22,11)~~ ~~(10,10,23,11)~~	(11,15,20,11) (12,15,21,11) ~~(13,15,22,11)~~	(1,14,8,10) ~~(2,14,9,10)~~ ~~(3,14,10,10)~~	(1,14,11,15) (2,14,12,15) ~~(3,14,13,15)~~	-

$$L_p = \{(28,12,39,8), (29,12,40,8), (20,11,28,12), (21,11,29,12), (8,10,21,11), (11,15,20,11), (12,15,21,11), (1,14,8,10), (2,14,12,15)\}$$

Figure 5.1: An example of forward and backward movements for a participant with $OS_p = 14$, $DS_p = 3$, $\Delta t = 1$ min, $TW_p = [1, 40]$, and $T_p^{TB} = 40$ min

of links that exist in at least one driver's set of links, since a rider needs to be carried by a driver on all the links that from his/her route.

Note that the ellipses that form boundaries of the reduced networks strictly prevent any feasible stations from being cut off. Furthermore, the link generation procedure uses conservative estimate of the traveling speed in the region to ensure that no feasible links are cut off. Therefore, the link set L can be safely replaced by $L_p, \forall p \in P$. Furthermore, since the potential drivers for each rider are determined based on the reduced networks and the forward and backward movements, no potential driver is excluded from the set of eligible drivers for each rider. Hence, use of the refined input sets generated by ESTAM in the ride-matching problem does not affect the optimal matching of riders and drivers.

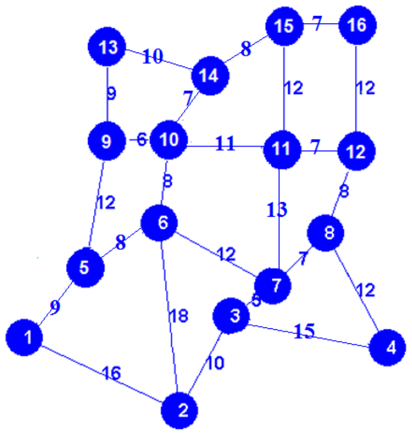
Chapter 6

The Many-to-One Matching Problem

The simplest from of a multi-hop matching problem, in which a single rider can travel by transferring between drivers, is the many-to-one problem. This type of matching is useful for systems that serve riders on a first-come first-served (FCFS), or any other pre-arranged order. This matching problem is a special case of the many-to-many problem presented in Chapter 4. In this chapter, we propose a Dynamic Programming (DP) algorithm that can solve this problem to optimality in real-time.

6.1 Illustrative Example

Throughout this chapter, we use the following problem instance to illustrate the concepts and demonstrate different steps of the proposed algorithm. This example contains a network with 16 stations, one rider, and four drivers. The network and the characteristics of the participants are presented in Figure 6.1. It is assumed that participants have no restrictions on the people with whom they travel.



Participant	O_p	D_p	T_p^{ED}	T_p^{LA}	T_p^{TB}
r_1	5	12	2	39	37
d_1	7	12	3	20	17
d_2	14	5	3	28	25
d_3	3	14	4	38	34
d_4	1	2	10	29	19

Figure 6.1: Problem instance used for demonstrating the steps of the DP algorithm throughout the chapter

6.2 Preprocessing using ESTAM

As described in Chapter 5, ESTAM generates for each rider a time-expanded feasible network in which the rider's optimal itinerary could be searched. This network is constructed from the links on the rider's reduced time-expanded network that are reachable by drivers. In this section, we review the three steps of ESTAM for the example presented in section 6.1, and generate the time-expanded feasible network for the rider in our example. In the next section, we use the DP algorithm to search on this time-expanded feasible network for the rider's optimal itinerary.

Figure 6.2 displays the reduced networks for the participants in our example. r_1 is planning to travel from station 5 to station 12. G_r is pointed out in Figure 6.2 as the section of the network confined inside the dashed green ellipse. The reduced networks for the four drivers in our example are also demonstrated in this figure, using solid red ellipses. The shaded area shows the intersection of the rider's reduced network with those of the drivers. This figure demonstrates that the rider's origin and destination stations are both inside the shaded region (i.e. reachable spatially by drivers), and so this rider has a chance of being served. If either of the origin or destination stations were not located in the shaded region, we could immediately conclude that the rider could not be served. For example, if driver 2 (traveling from station 14 to 5) was not in the set of drivers, the rider could not be served, due to lack of space proximity at its origin station (station 5).

Figure 6.2 can also help us identify and eliminate drivers who cannot contribute to the rider's itinerary. As an example, driver 4 in Figure 6.2 can be filtered out from the problem, reducing set $D = \{d_1, d_2, d_3, d_4\}$ to $D = \{d_1, d_2, d_3\}$. Driver 3 does not have proximity to the rider's origin or destination, but can contribute to portions of the travel in case of a transfer. In fact, in this particular example, the only way the rider can travel from station 5 to station 12 is by being picked up by driver 2, transferring to driver 3, and finally transferring to driver 1. The exact transfer locations, and deciding whether such a route is feasible at all under the participants' travel time restrictions will be examined later. For instance, it might be the case that by the time the rider reaches station 6, 9 or 10 by driver 2, driver 3 needs to have traveled beyond those stations to reach its destination on time (thereby, a spatiotemporal feasible path for the rider would not exist).

Next, let us look into the time-proximity, using the link generation procedure detailed in Chapter 5. Figure 6.3 displays the time-expanded reduced network for driver 2 in our example. Note that it is possible to have more than one origin and/or destination node in a time-expanded reduced network. Recall that each node is represented by a 2-tuple of a time interval and a station. Nodes in the form of $(t, OS_p), \forall t \in T_{OS_p}$ are all origin nodes, and nodes in form of $(t, DS_p), \forall t \in T_{DS_p}$ are all destination nodes. We keep these nodes in origin and destination sets, $O(p)$ and $D(p)$ respectively. In Figure 6.3, $O(d_2) = \{(3, 14), (4, 14)\}$, and $D(d_2) = \{(27, 5), (28, 5)\}$.

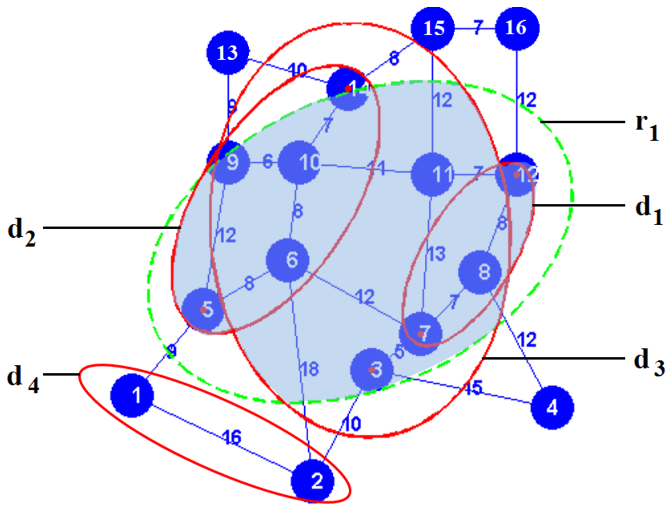


Figure 6.2: Reduced networks of all the participant in our example

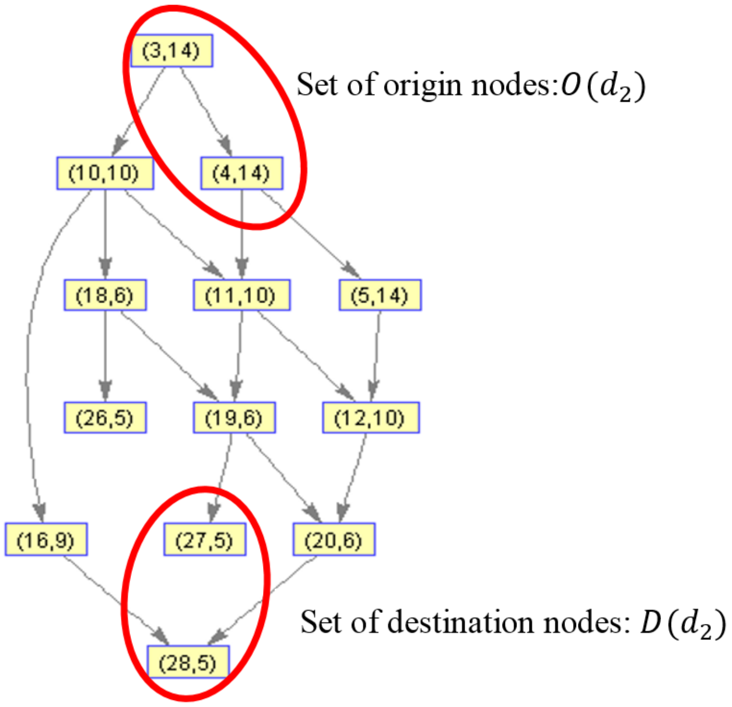


Figure 6.3: Time-expanded reduced network for driver 2 in Figure 6.2

Finally, Figure 6.4a shows the time-expanded feasible network for the rider in our example.

6.3 Solution Methodology

The ESTAM algorithm constructs the time-expanded feasible network for the rider in our problem, on which the rider's optimal itinerary can be searched. Even though this network is substantially smaller than the original network, the number of feasible routes for the rider can still be very large. Different combinations of links on which to travel, and drivers with whom to ride, can make enumerating the paths computationally prohibitive (see Figure 6.4b). In this section, introduce an algorithm using which the optimal path for the rider can be found quite efficiently.

Figure 6.4a shows the time-expanded feasible network for the rider in our example. The goal is to find an optimal itinerary for the rider to travel from his/her origin node to his/her destination node. We define the origin node as the node in the rider's origin set , O_r, with the smallest time interval, and the destination node as the node in the rider's destination set, D_r, with the largest time interval. In our example, the origin node is $(3, 5)$, and the destination node is $(29, 12)$.

The optimal itinerary as defined in this study is a path that minimizes a linear combination of the in-vehicle travel time, wait time, and the number of transfers. Before searching for the optimal path, two preliminary operations need to be performed on the graph. First, nodes that are not descendants of the origin node, or predecessors of the destination node should be removed. An example of this operation is demonstrated in Figure 6.4a. This figure displays the time-expanded feasible network of the rider in our example who is traveling from station 5 to station 12. The components of the graph containing nodes $\{(22, 7), (23, 7), (24, 7)\}$, and $\{(14, 9), (15, 9)\}$ (marked by red rectangles in Figure 6.4a) have to be eliminated from the graph, since they are not descendants of the origin node $(3, 5)$. This task can be easily accomplished by scanning the adjacency matrix of the graph.

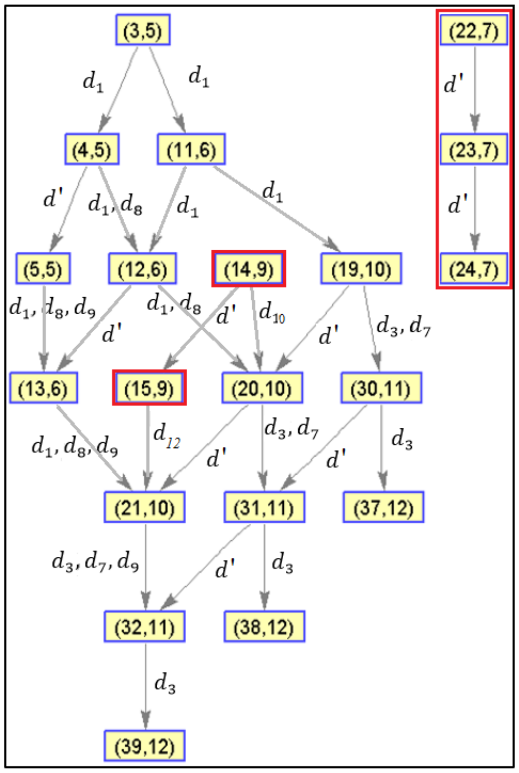
The second operation is to topologically sort the graph. A topological ordering for such a graph always exists, since the graph is directed and acyclic (the graph is acyclic since nodes have a time component, and the link generation procedure does not generate links that represent traveling back in time). Figure 6.4b displays the resulting graph after performing these two preliminary operations. The numbers to the left of nodes in this figure show the topological orders. The numbers on the links show the IDs of the drivers who can carry the rider on the links.

Doing a depth first search (DFS) on this revised time-expanded feasible network (Figure 6.4b) determines whether an itinerary for the rider exists at all. Starting from the first node in the topologically ordered graph, if DFS does not conquer a node in the destination set, then the rider cannot be served in the system. Otherwise, if DFS finds a path (establishing that a matching is spatially and temporally feasible), we will use the proposed DP algorithm to find the best path for the rider.

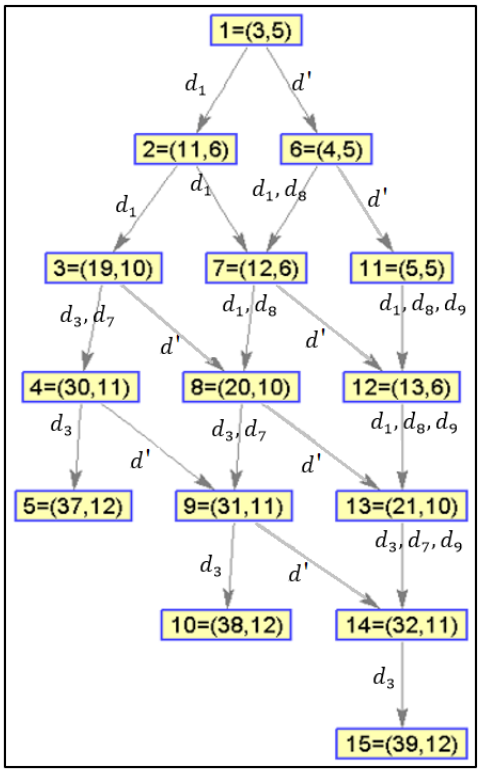
6.3.1 The Dynamic Programing (DP) Algorithm

Let us start by defining sets D_j^{IN} and D_j^{OUT} as the set of drivers who enter and exit node j respectively, and set DN_j for each node j as a set of tuples (i, d) such that there is a link for driver d from node i to j in the revised time-expanded feasible network. Furthermore, we denote by $V(j, d)$ the minimum cost of the optimal path from the start node (node 1 in the topologically ordered graph) to node j, such that the last link on the path is traversed by driver d. The goal is to find $V_r^* = min_{j \in D(r), d \in D_j^{IN}} \{V(j, d)\}$.

We initialize the $N_r \times D_r$ matrix $V(\cdot)$ to infinity, where N_r is the number of nodes in rider r's



(a) Time-expanded feasible network



(b) Revised time-expanded feasible network (after performing the preliminary operations)

Figure 6.4: Time-expanded feasible network for the rider in our example before and after revision

revised time-expanded network, and D_r is the number of drivers that contribute to rider r's revised time-expanded feasible network. To keep track of the optimal solution, we introduce two additional matrices of the same size, called $Pred - Node$ and $Pred - Driver$, and initialize them to zero. These matrices keep the predecessor node and the driver that takes the rider to the predecessor node on the optimal path, respectively.

We set the initial condition as $V(1,d) = 0$, for all $d \in D_1^{OUT}$. In the revised time-expanded feasible network in Figure 6.4b, the initial condition is $V(1,d_1) = 0$. After setting the initial condition, we traverse the nodes in the (topologically ordered) graph in ascending order. For each node j, and each driver d in the set D_j^{IN}, $V(j,d)$ is computed as in equation (6.1) below:

$$V(j,d) = min_{i:(i,d)\in DN_j}\big\{min_{d'\in D_i^{IN}\setminus ED(i,d')}\{V(i,d') + C_T 1_{\{d\neq d'\}}\} + C(i,j)\big\} \tag{6.1}$$

The indicator function ($1_{d\neq d'}$) returns 1 if $d \neq d'$ (i.e., if the rider makes a transfer) and returns zero otherwise. C_T is the penalty (cost) of a transfer. The function $C(i,j)$ gives the cost of traveling on link $(i,j) = (t_i, s_i, t_j, s_j)$, and can be calculated using equation 6.2. The cost of travel as defined in equation (6.2) is the travel time on the link in case a travel takes place (i.e., $s_i \neq s_j$), or a penalty W if the rider waits at the station ($s_i = s_j$) for one time interval. Note that if the rider stays in the same station for multiple time intervals, the algorithm will automatically accumulate W the appropriate number of times. This is due to the fact that the same station at different time intervals corresponds to separate nodes on the time-expanded feasible network. Further, note that $C(i,j)$ in equation (6.2) can be easily extended to $C(i,j,d)$, in case a more customized driver-dependent cost is desired. We discuss the set $ED(i,d')$ used in equation (6.1) further down.

$$C(i,j) = \begin{cases} (t_j - t_i + 1)dt & s_i \neq s_j \\ W & s_i = s_j \end{cases} \tag{6.2}$$

After each computation of $V(j,d)$ using the recursive function in equation (6.1), we save $n_i^* = argmin_{i:(i,d)\in DN_j}\big\{min_{d'\in D_i^{IN}\setminus ED(i,d')}\{V(i,d')+C_T 1_{\{d\neq d'\}}\}+C(i,j)\big\}$ in $Pred-Node(j,d)$, and $d^* = argmin_{d'\in D_i^{IN}\setminus ED(i,d')}\{V(i,d') + C_T 1_{\{d\neq d'\}}\}$ in $Pred-Driver(j,d)$. Once the optimal minimum cost for the rider is found, these matrices can be used to retrieve the corresponding optimal itinerary.

We define set $ED(i,d')$ to include all drivers on the optimal path to node i, excluding the driver on the last link (d'). Drivers in this set are excluded from set D_i^{IN} in equation (6.1) to ensure the feasibility of the solution. Although very unlikely, it could happen that a route includes links that cannot co-exist. If a driver covers multiple links on a rider's itinerary, and those links belong to different feasible paths of the driver, then the resulting itinerary is not feasible. To clarify, an example is shown in Figure 6.5. In this example, driver d has two potential paths to make his/her trip, while driver d' has only one. There is only one itinerary for the rider, marked by dashed red arrows. This itinerary consists of three links. The first link is covered by driver d via this driver's first potential path, the second link is covered by d', and the third link is covered by d via this driver's second potential path. Clearly the resulting itinerary is not feasible, since d cannot carry the rider on both links. Excluding drivers in set ED from the list of potential drivers in equation (6.1) ensures that such an itinerary is never generated. In the example in Figure 6.5, the rider cannot be served.

Using the proposed DP algorithm, not only the optimal itinerary, but all the feasible itineraries for a rider are readily available. Therefore, if the solution that is deemed optimal based on the objectives of the system (i.e. number of transfers, and traveling and waiting times) did not satisfy other personal requirements of participants, we can easily access and examine the next best solutions.

Once a rider's itinerary is fixed, it is easy to find the optimal itineraries for drivers. If a driver is part of a rider's itinerary, then a portion of his/her path is fixed. Given the fixed portions, it is easy to find the shortest travel time path for the driver from the driver's

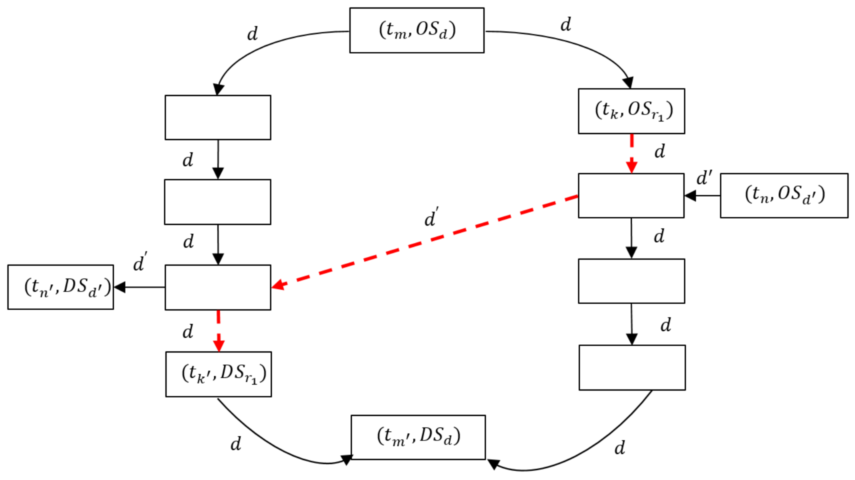


Figure 6.5: Example of an infeasible solution

origin station to his/her first fixed portion, between the fixed portions, and from the last fixed portion to the driver's destination station. Notice that because the fixed portions of a driver's path have time components, it is easy to order them. After these ordering is done, Dijkstra's algorithm can be used to find the shortest path between the fixed stations in the driver's reduced network.

6.4 Numerical Experiments

In this section, we generate multiple random instances of the ridesharing problem in a network with 49 stations. All problem instances contain 1000 participants. We vary the ratio of drivers and riders in each problem instance, and assess the performance of the proposed ridesharing algorithm as the ratio of riders in the problem changes. The participants' earliest arrival times are generated uniformly within a one hour time period.

All drivers are assumed to have four empty seats, and riders are assumed to accept up to 3 transfers. We use 1 min time intervals, and consider a randomly generated maximum ride time within the interval $[T_p^{ST}, 1.1T_p^{ST}]$ for participant $p \in P$, where T_p^{ST} is the shortest path travel time between the participant's origin and destination stations. The latest arrival time for each participant is computed as the sum of their earliest departure time and maximum ride time. In the interest of simplicity, we assume that participants have no personal requirements on the people with whom they travel.

In the next two sections, we use the proposed algorithm to find the optimal solutions for the randomly generated problems through simulations. In section 6.4.1, the origin and destination locations of participants will be selected based on a uniform random distribution. This type of selection serves as the default setting in our simulations. In section 6.4.2, we study a more practical scenario, where trip ends are determined randomly, but based on a clustering of the network.

6.4.1 Uniform Random Selection of Trip End Locations

For the problem instances generated in this section, the origin and destination stations of participants are selected from the 49 stations, based on a uniform random distribution. The results we report in this section are averaged over 10 simulation runs for each problem. Figures 6.6a and 6.6b display the solution times, and the percentage and number of served riders respectively.

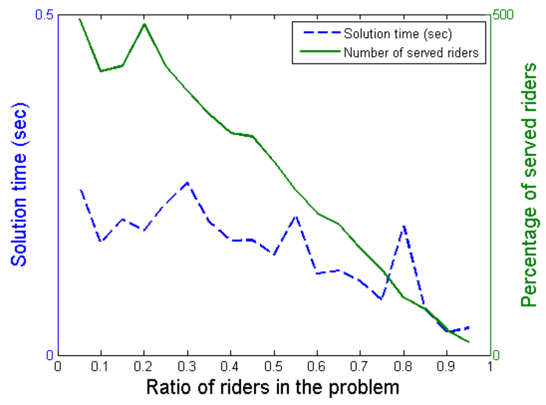
As mentioned before, riders are considered in a FCFS basis in simulations, and once a rider registers in the system, a ride-matching problem is solved for that rider (i.e. the notification deadlines of participants are assumed to be their registration times). The solution time reported for each problem instance in Figures 6.6a and 6.6b is the highest solution time among all riders in the problem (averaged over 10 runs).

Figure 6.6a shows that the percentage of served riders decreases as we increase the ratio of riders in the problem. Note that since the number of participants is held constant (1000), by increasing the number of riders we are in fact decreasing the number of drivers. Since this rise on the demand side is met with a decline on the supply side, naturally, a declining trend in the matching rate is witnessed.

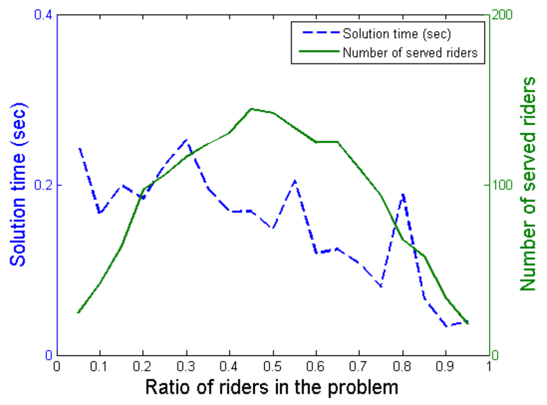
Figure 6.6b suggests that the number of served riders is maximized when the number of riders in the system is slightly lower than number of drivers, i.e. the ratio of drivers to riders is slightly higher than 1. At his best case scenario (highest number of riders served), about 30% of the demand for rides is satisfied. If the interest of the system is to maximize the percentage of served riders, about 50% of riders can be served in the best case scenario. Notice that this higher performance is obtained when the number of riders in the system is only 50, and there are 950 drivers.

Figure 6.6c shows the number of matched participants in the system. This figure suggests that the number of matched participants is maximized in the same range where the number of served riders is maximized. In fact, when riders constitute about 45% of the total number of participants, the number of matched riders and drivers are both maximized. Notice that this is not an obvious conclusion, since each driver can carry multiple riders, and each rider can transfer between multiple drivers. Furthermore, notice that these results are only valid for the case where the origin and destination locations of trips are randomly selected. We show in the next section that in more practical settings where business districts and residential areas are separated (and therefore trip ends have higher spatial proximity), the number of matched participants can increase drastically.

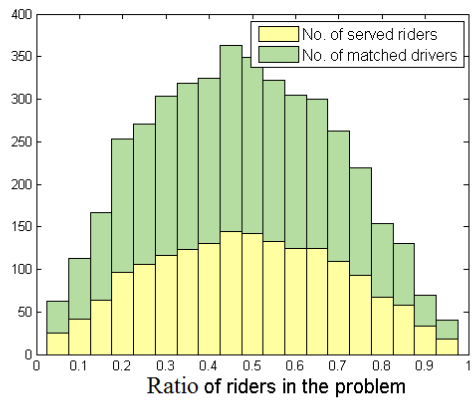
Finally, Figure 6.6d shows the distribution of number of transfers for each problem instance. This figure suggests that the majority of served riders do not have to make any transfers. This figure also suggests that the number of additional riders that can be served as a result of allowing transfers in the system is not trivial.



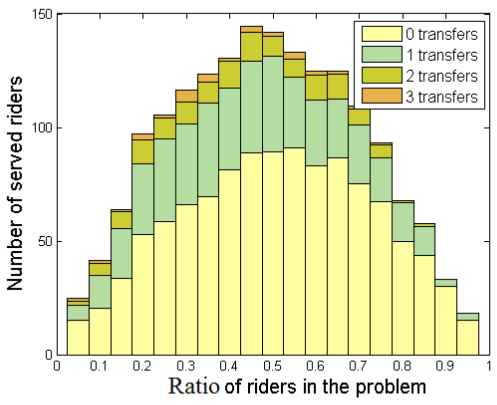
(a) Change in solution time and percentage of served riders with ratio of riders



(b) Change in solution time and the number of served riders with ratio of riders



(c) Distribution of number of participants matched in each problem



(d) Distribution of number of transfers in the optimal solutions

Figure 6.6: Algorithm performance for the randomly generated problem instances, averaged over 10 runs

6.4.2 Uniform Random Selection of Trip Ends in Clusters

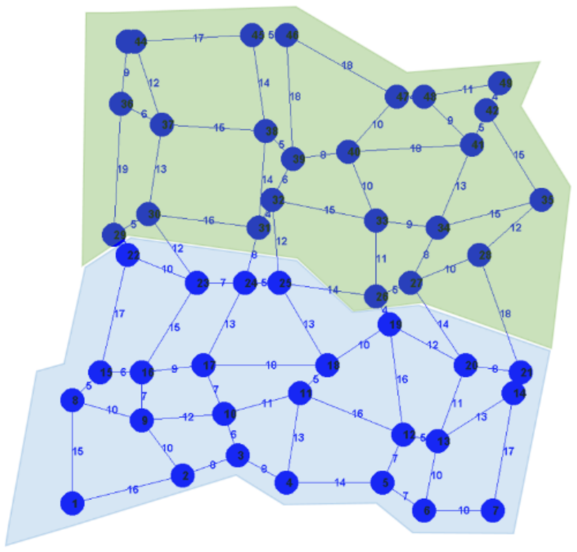
In the previous section, we made the assumption that the origin and destination locations of trips are completely random. In reality, residential areas are usually close to each other, and separate from commercial and business districts. In this section, we identify two geographically distinct regions on the network to represent residential and business districts. Similar to the previous section, we generate different problem instances, varying the percentage of riders in the problems. For each problem instance (each problem instance having a different ratio of riders), we conduct 10 simulation runs, and report the average results.

Figure 6.7a shows the clustering of a sample network. The area on the top is considered to be the business district, and the area on the bottom is assumed to be the residential area. Each problem instance represents the simulation of a morning peak period, in which participants travel from a randomly selected station in the residential area, to one in the business district.

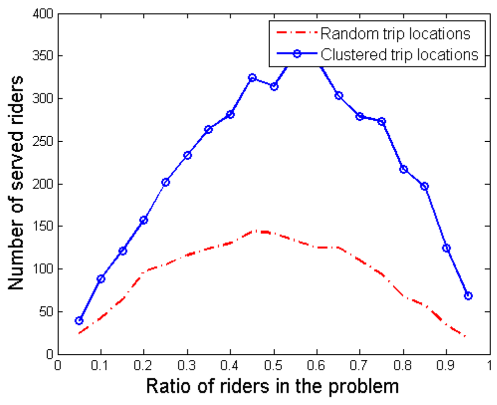
Figure 6.7b shows that number of served riders has the same trend compared to the scenario where origin and destination locations were completely random, but in this case the number of served riders has more than doubled. Figure 6.7c suggests that in this more practical setting, when the system has the optimum ratio of drivers and riders, more than 60% of the participants will actually participate in the system.

6.4.3 The Critical Mass

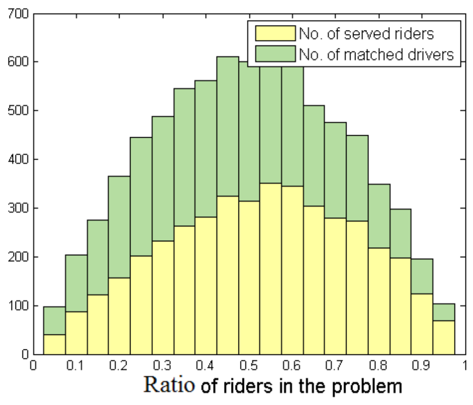
For a ridesharing system to be able to work independently, a critical mass of participants is required. In the previous sections, we demonstrated the performance of a ridesharing system with 1000 participants. We showed that in the more practical scenario of having distinct business and residential areas, more than 60%, and in the completely random case, about



(a) A sample clustering of the network



(b) Comparison of number of served riders between the completely random scenario and the clustering scenario



(c) Distribution of matched participants in the clustering scenario

Figure 6.7: System performance for the scenario with distinct business and residential areas. Results are averaged over 10 runs.

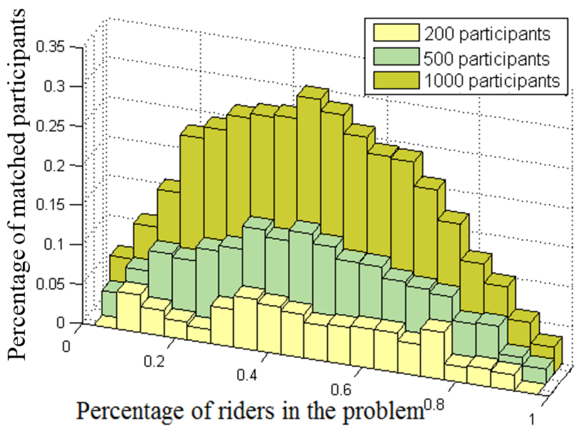


Figure 6.8: Sensitivity analysis over the number of participants in the system

35% of participants can be successfully matched. It is intuitive to expect the average number of successful matches to increase with the number of participants. However, for a system to survive, a minimum number of participants should be able to use the system successfully. This success will encourage users to consider participating in the system again, and promote the system to their friends and family members.

In this section, we do a sensitivity analysis on the number of participants in the system. This analysis can shed light on the relationship between system performance, and the number and ratio of participants.

Figure 6.8 shows the percentage of matched participants as the total number of participants changes from 200 to 1000 individuals. Similar to the previous sections, for a given number of participants, we generate multiple random problem instances by changing the ratio of riders. The travel ends of participants are selected at random with uniform probabilities. Figure 6.8 suggests that the increase in the number of participants leads to higher system performance, in terms of the percentage of matched participants. The peak performance of the system in all cases, however, occurs in the same range.

6.4.4 P2P Ride Exchange

Ridesharing systems are in general spatiotemporally sparse. The temporal sparsity stems from the rather tight time windows of participants, and the spatial sparsity is due to the participants' fixed origin and destination locations. This spatiotemporal sparsity limits the number of satisfied requests. In fact, one of the contributors to the initial failure of P2P ridesharing systems in the US in the 1990s was the very small number of served rider requests.

Today, advancements in the communication technology and prevalence of smartphones, along with the young generation's fascination with use of technology has led to a larger pool of riders and drivers interested in ridesharing. However, as discussed in the previous sections, a critical mass of participants is still an important requirement for ridesharing systems to ensure that they can operate independently, without having to outsource drivers. Therefore, a ridesharing system has to try and make the best use of its limited supply resource, the drivers.

A ridesharing system that takes riders into consideration only one at a time, is in fact wasting its very limited and valuable resources by fixing itineraries of matched drivers. Figure 6.9 shows an example of how fixing drivers' itineraries can deteriorate the performance of the system. This example includes two riders and two drivers. Assume rider 1 has registered in the system before rider 2, and hence the system starts by finding a match for rider 1. Vehicles 1 and 2 are both candidates to be matched with this rider. Eventually the system matches rider 1 with driver 2, since driver 1 has to make a larger detour to carry the rider. Driver 2's itinerary, marked in solid blue line, is then fixed. The next rider in queue is rider 2. This rider could have been matched with driver 2 through the route marked with the blue dashed line, if driver 2's itinerary had not been fixed. However, under the current state of the system, rider 2 cannot be matched. The ridesharing system, therefore, can only match one rider with one driver.

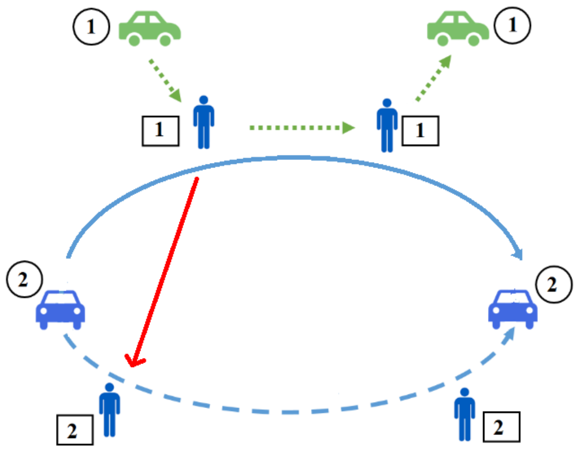


Figure 6.9: P2P ride exchange

In the example in Figure 6.9, we could have increased the system performance by solving a ride-matching problem that included the two available riders (using the formulation in Chapter 4), as opposed to considering them one at a time. The solution to such a problem would match rider 1 with driver 1, and rider 2 with driver 2, serving both riders, and involving all 4 participants.

There are two issues that may prevent us from the benefits of solving a many-to-many matching problem. First, if rider 2 has not joined the system yet at the time when we have to notify rider 1 of the results, then solving the many-to-many problem would not be beneficial (we would obtain a solution similar to that of a one-to-many problem). Second, solving the many-to-many problem could be too time-consuming. Even efficient algorithms (as we'll see in Chapter 7) can take minutes to solve, even for moderately sized problem instances.

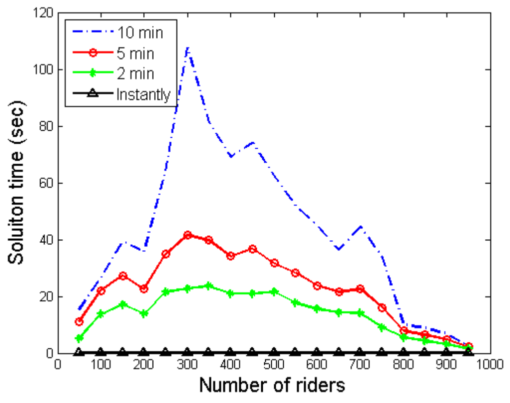
To demonstrate the difference between a many-to-many ride-matching problem (which includes multiple riders and multiple drivers) and a many-to-one problem (which includes one rider and multiple drivers) in terms of solution time and matching rate, we solve the problem instances in section 6.4.1 using both approaches. For the many-to-many problem, we use the decomposition algorithm proposed in Chapter 7 on a rolling time-horizon basis, by re-optimizing the system at 2, 5, and 10 minute time intervals. In this approach, instead of matching one rider at a time, we solve a problem that includes all the riders whose notification deadlines lie within the mentioned time interval. Figure 6.10 depicts the results of this comparison.

Figure 6.10 suggests that the solution time for a many-to-many ride-matching problem is too high for real-time applications. On the other hand, the number of matched riders improves when a many-to-many problem is solved. In this section, we propose P2P ride exchange as a mechanism to improve the solution of a many-to-one ride-matching problem, while still being able to serve riders in real-time. We elaborate on this mechanism in Chapter 8.

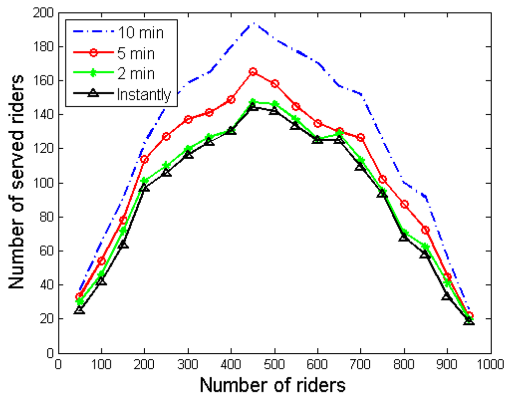
Although we can use the proposed algorithm in this chapter to solve a many-to-one ride-matching problem to optimality, the formulation of the problem itself is not optimal. P2P ride exchange can help us move from the sub-optimal solution obtained by matching one rider as a time, toward the optimal solution that can be obtained by including all riders in the matching problem, without the unattractive side-effect of the increased solution time.

In this approach, we still solve the matching problem for one rider at a time, and on a FCFS basis. However, we do not fix the itineraries of matched drivers. If a rider can be served only using a driver that has been previously matched (but though a conflicting itinerary), then the two riders can start a negotiation to make a ride exchange. Note that it is easy for the system to propose alternative itineraries to riders involved in the exchange, since all the feasible itineraries for riders are generated using the DP algorithm (see section 6.3).

In the example in Figure 6.9, if we solve the ride-matching problem for rider 2 without fixing the itinerary of driver 2, the solution will have driver 2 routed on the dashed path, and match it with rider 2. However, driver 2 was previously routed differently, and assigned to rider 1. These two riders can now engage in a negotiation. Since rider 1 can also be matched with driver 1, there is a good chance that rider 1 will accept the proposed ride exchange



(a) Distribution of solution times



(b) Distribution of number of matched participants

Figure 6.10: Solution times and number of served riders for each randomly generated problem instance under different re-optimization periods

by rider 2, in exchange for money, or credit toward using the ridesharing system. In case the negotiation is successful, the final matching would be equal to the matching obtained by solving a many-to-many ride-matching problem.

In addition to the benefits that P2P ride exchange can offer in terms of system performance, it can also make riders more engaged in the system. Riders can earn money or credit by selling their itineraries, and settling for less efficient itineraries suggested by the system. However, for such a system to work properly, a good mechanism should be designed to ensure that individuals cannot take advantage of the system by providing untruthful information.

6.4.5 Overlapping Sets of Drivers and Riders

Initially, we made the assumption that riders and drivers form two mutually exclusive sets. In this section, we study the scenario in which participants who register in the system as riders might be doing so because they prefer traveling as riders, and not because they do not own or have access to a vehicle. If the system is not able to serve these participants as riders, then they will drive their own vehicles, and join the system as drivers. An extreme case would be to study the case where all riders have access to vehicles, and are willing to join the system as drivers. This case shows the maximum benefits a ridesharing system can offer.

In this section, we study the impact of this assumption on the problem instances in section 6.4.1. Figure 6.11 shows the maximum number of served ride requests under different percentage of riders who are willing to act as drivers. In the case of 0%, the problem becomes equivalent to the problem of a system with two mutually exclusive sets of riders and drivers. As this percentage increases, the shape of the curve starts to change. The best performance of the system can be obtained when 100% of riders have access to vehicles, and are willing to join the system as drivers, if they cannot be served as riders. It is interesting to see that in this best case scenario, the number of served riders has an increasing trend.

In the basic scenario with two mutually exclusive sets of riders and drivers, number of served riders reaches its peak when riders constitute around 45% of the participants. As the ratio of riders increases, the number of served riders decreases, because there are fewer number of drivers available. If riders who cannot be matched join the system as drivers, this practically increases the number of drivers in the ranges where previously there was a shortage of drivers, and this leads to higher number of riders being served. Figure 6.11 suggests that encouraging such a strategy in a ridesharing system can drastically improve its performance.

6.5 Case Study: P2P Ridesharing as Transit Feeder in the Los Angeles County

One of the main issues faced by major cities in the US today is congestion. In addition to directly impacting travelers by increasing travel time and reducing travel time reliability, congestion leads to higher levels of Green House Gas (GHG) emissions which are harmful to

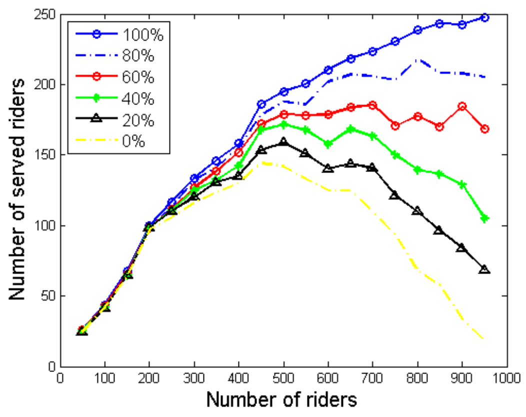
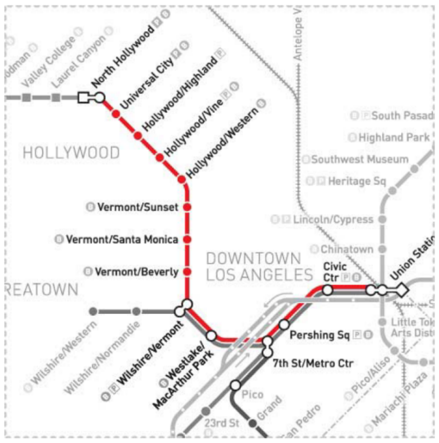


Figure 6.11: Number of served riders under different percentage of riders (who can switch and become drivers)

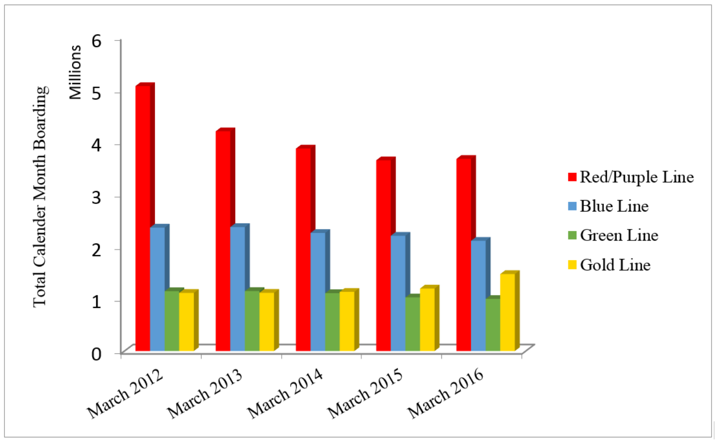
the environment and people's health. One of the solutions as to how to reduce congestion is to eliminate vehicles from roads by putting individuals who are traveling along the same routes in the same vehicles.

In recent years, thanks primarily to increase in carpooling ridesharing in the US has experienced a slight increase in mode share, reaching 11% in 2008. Although this increase in ridesharing seems to be a step forward in the direction of a greener transportation system, this is not necessarily the case. The kind of modal shift that occurred as a result of the introduction of ridesharing is as important. The benefits of ridesharing depend tremendously on the nature of the modal shift. The benefits would be high in terms of reducing congestion and GHG emissions if the demand is being shifted from single-occupancy vehicles to rideshare systems, but may not be significant if the ridesharing demand is being shifted from transit. In addition, introduction of ridesharing can lead to emergence of more complex multi-modal alternatives, such as the transit-rideshare mode.

Study of the many government-funded ridesharing systems indicate that ridesharing systems, as they work today, are competitive to transit systems (Levofsky and Greenberg (2001); Nelson Nygaard Consulting Associates and RideNow Inc (2006)). The goal of this section is to assess the potential of ridesharing in being a complement to transit, feeding it instead



(a) Los Angeles Metro red line



(b) Ridership trends of LA metro

Figure 6.12: Case study area: Los Angeles Metro red line

of shifting demand away from it. We analyze the potential of such multi-modal travel using a parametric study with simulated demand based on the current Southern California Association of Governments (SCAG) model of a selected area (LA Metro red Line catchment area).

The reason why we consider the LA Metro red line (Figure 6.12a) to study potential rideshare based demand augmentation is the noticeable reduction of ridership (Figure 6.12b) in recent years. Such a trend indicates potential opportunities for additional demand-inducement strategies using rideshare options.

The success of a multi-modal transit-rideshare system can be considerably influenced by the architecture of the designed system, namely locations where the ridesharing service is offered, price of ridesharing, and the matching method used by the system. In this section, we elaborate on this system architecture, and show case the impact of such targeted architecture on transit ridership augmentation for the LA Metro red line.

6.5.1 Preliminaries

We use the DP algorithm introduced in this chapter for routing passengers. Transit lines can enter this model as drivers with fixed routes. Here, we elaborate on how we have obtained other sets of input for the DP algorithm, with main focus on set of stations and links.

Stations

In order to avoid confusion between ridesharing and transit stations, here we refer to ridesharing stations as "go-points". Furthermore, due to the large size of the network, we only treat a subset of go-points as locations were transfers can occur, and call such points "transfer points". Each go-point belongs to a neighboring transfer-point. Individuals can travel directly between the go-points attributed to the same transfer point. However, in order to travel between two go-points attributed to two different transfer points, an individual has to travel between the two transfer points. Note that despite traveling between transfer points, a transfer does not have to occur, i.e. an individual can enter and exit a transfer point in the same vehicle.

The SCAG region has a total of over 4000 TAZs (i.e. 16 million OD pairs). Our goal is to identify significant OD pairs in terms of demand level, and use this information to identify the go- and transfer points in the network. For this purpose, we identified OD pairs in the SCAG trip tables with hourly trip rates higher than 10. We limited our analysis to single-occupancy auto demand only, because the focus of the study is to identify potential modal shift from drive-alone to rideshare and rideshare-transit alternatives.

We select a subset of go-points that fulfill two criteria as transfer points. Transfer points should be distributed in the network such that (i) they are located closer to go-points with higher levels of demand, and (ii) they are distributed in the network as evenly as possible.

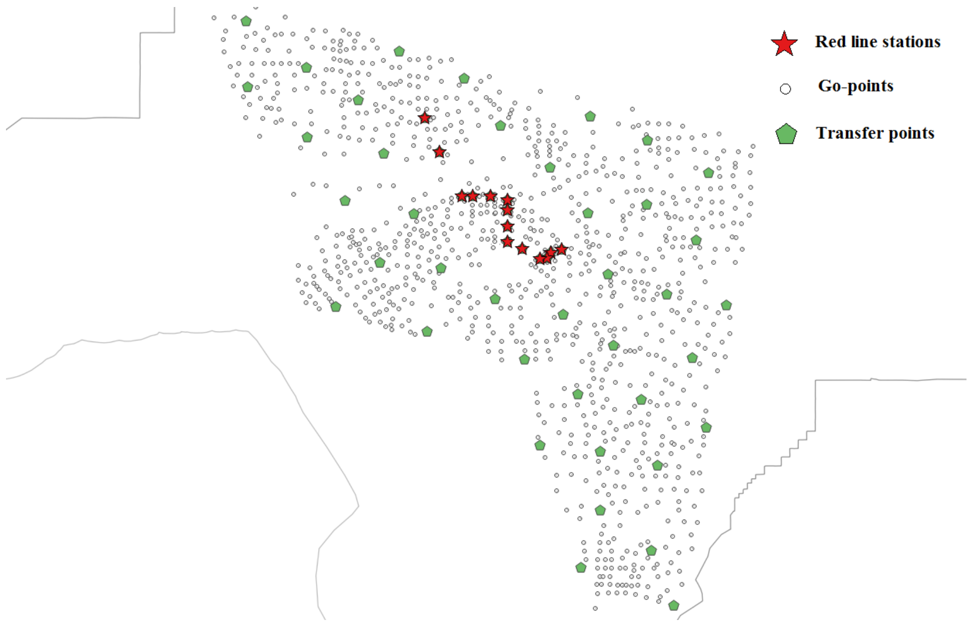


Figure 6.13: Three types of stations in the LA network

Red line stations, go-points, and transfer points are displayed in Figure 6.13.

Link Sets

We introduce three families of link sets that connect different types of stations (i.e. go-points, transfer points, and red line stations) in the network. Figure 6.14 demonstrates the three families of link sets. The first link set displayed in Figure 6.14a connects transfer points to each other. The second link set (Figure 6.14b) connects go-points to their corresponding transfer points. Figure 6.14c demonstrates the third link set that connects the red line stations to their nearby go-points. This link set connects each of the go-points confined within a 2.5 mile radius of at least one of the red line stations to the red line stations located within their 2.5 miles radius. Notice that we do not generate any ridesharing links that connect red line stations together, or to their nearby go-points, in order to have the itineraries use transit whenever possible.

Each go-point in the network is connected to at least one transfer point (link sets 2 and

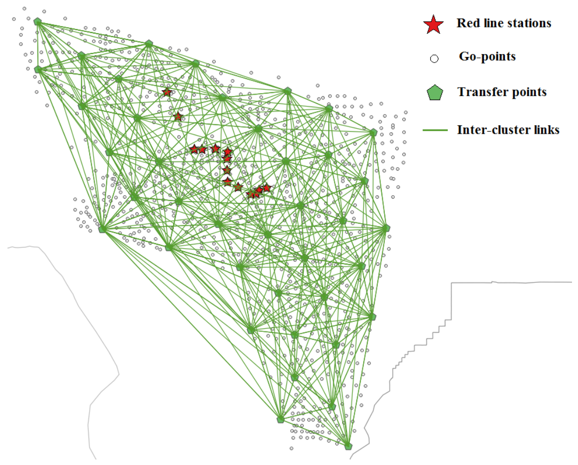
3). In addition, all transfer points are connected to each other (link set 1). This indicates that there is a path between any two go-points in the network. Link set 2 implies that the shortest path of a rider who is traveling between two go-points (that are not within a 2.5 miles radius of the red line) includes traveling to the transfer point corresponding to the origin go-point; next, traveling from this transfer point to the transfer point corresponding to the destination go-point; and finally traveling from there to the destination go-point itself. Another implication of link set 2 is that since all the go-points corresponding to the same transfer point are connected to each other, if a rider needs to make a short trip between two such go-points, no transfers are required.

For practical reasons, it is assumed that transfers for trips that originate from or are destined to go-points within a 2.5 mile radius of the red line stations are limited to the metro red line stations only. Hence, link set 3 connects such go-points to the red line stations directly. This link set is appropriate to use, because we wish to promote the rideshare-transit option. In the contrary, if the goal was for ridesharing to replace transit, we could introduce links that connect such go-points to each other, rather than to the transit stations.

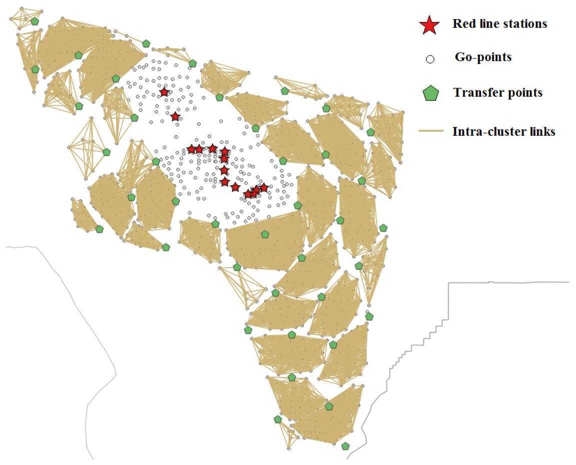
6.5.2 Results

In this section, we study the modal shift from drive-alone to rideshare and rideshare-transit alternatives using simulations. Simulations are done for the morning peak hour in LA. Origin-destination trip tables used in simulations are obtained from the SCAG planning model, and spread throughout the three-hour morning peak period based on a uniform distribution.

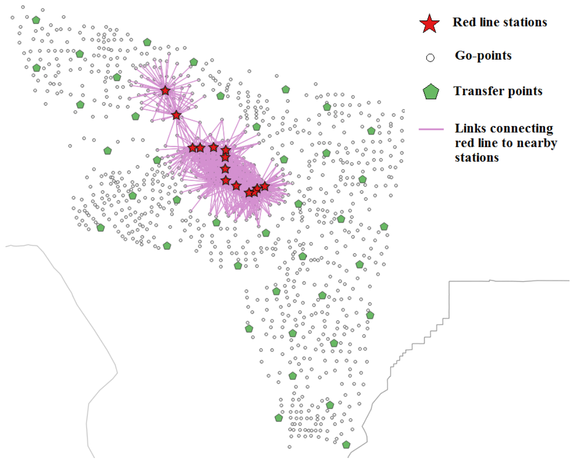
For each simulation run, we randomly select our set of riders and drivers. In all simulation runs, we use 1,000 riders, but change the number of drivers from 1,000 to 160,000 in order to study the impact of the rider to driver ratio on the matching rate. We assume each vehicle has the capacity to carry 4 passengers, and do not a set a limit on the number of transfers.



(a) Link set 1: Links connecting transfer points to each other



(b) Link set 2: Links connecting go-points to their corresponding transfer points



(c) Links connecting red line stations to the nearby go-points

Figure 6.14: Link sets

In addition, we assume that maximum ride times for participants are 20% higher than their shortest path travel times.

The cost function we use in the DP algorithm is the sum of four components: (i) a distance-based fare, (ii) dollar value of travel time, (iii) dollar value of additional penalty for waiting time, and (iv) dollar value of penalty for transfers. We consider a default value of \$20/hr for value of time (VOT), \$0.25/mile distance-based fare, \$1.5 fare for use of transit, \$0.1 as the monetary equivalent of additional penalty for waiting for each time period (in addition to the value of time), and \$0.1 as the monetary equivalent of the penalty for each transfer.

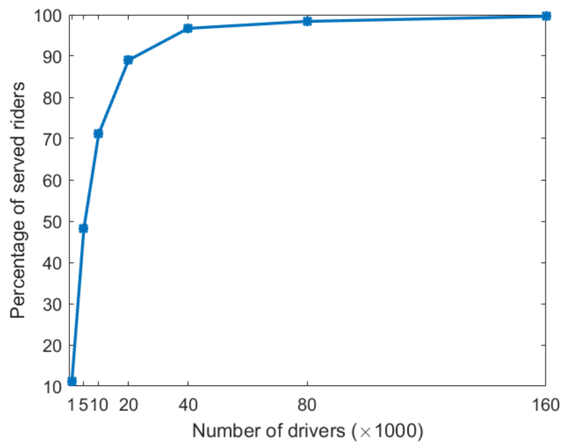
Matching Rate

In order to study the impact of the number of drivers on the matching rate, we ran a set of simulations. The resulting matching rates are displayed in Figure 6.15.

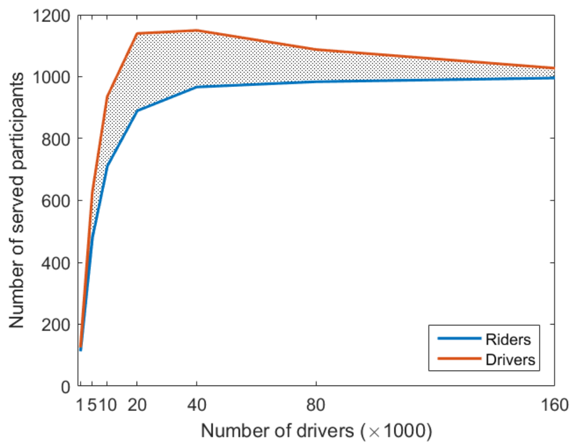
Figure 6.15a displays the percentage of served riders as a function of the number of drivers. As intuition suggests, this percentage increases with the number of drivers. Percentage of served riders, however, grows with the number of drivers at a rate slower than linear. For example, with a 5,000 unit increase in the number of drivers (going from 5,000 to 10,000 drivers), we witness a 20% increase in the percentage of served riders. However, in order to experience another 20% increase in the percentage of served riders, we have to have a 10,000 unit increase in the number drivers (going from 10,000 to 20,000 drivers).

Figure 6.15b displays the number of served riders and matched drivers as a function of the number of drivers in the system. This figure sheds light on the performance of the system under different levels of supply (i.e. number of drivers).

When the number of drivers is at its lowest, the number of served riders is about the same as the number of matched drivers, implying that most trips are being served without transfers



(a) Percentage of served riders



(b) Number of served participants

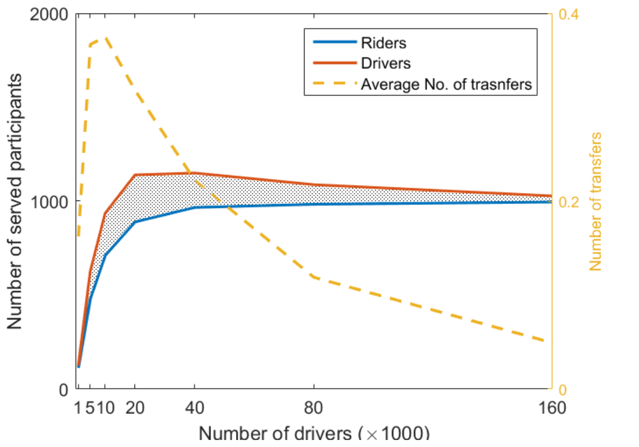
Figure 6.15: Matching rate as a function of number of drivers

(this conclusion is confirmed by looking at the number of transfers for each level of supply in Figure 6.16, as we will discuss in the following section). At such low level of supply, the number of drivers is too small for multi-hop routes to be formed for riders. Up to a certain level (20,000 drivers), the difference between the number of matched riders and drivers (i.e. the horizontal distance between the two curves in Figure 6.15b) keeps increasing. Finally, when the supply level becomes really large, and most of the demand is being served, there is no need for more costly multi-hop routes anymore, and the number of matched riders and drivers start to converge again.

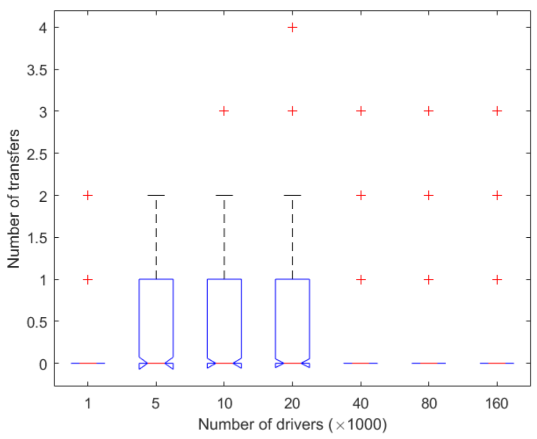
Number of transfers

Figure 6.16a demonstrates the number of transfers under different levels of supply. This figure suggests that when the number of drivers is too low or too high, transfers are very limited, and most riders can be served with zero transfers (Figure 6.16b). However, more transfers are required in the middle ranges.

As discussed in the previous section, at very low supply levels there are not enough drivers in



(a) Relationship between the average number of transfers and the number of served riders and drivers



(b) Boxplot of number of transfers

Figure 6.16: Number of transfers

the system to form multi-hop routes, and at very high supply levels almost all ride requests can be served without any transfers, and therefore an overwhelming number of trips end up being single-hop. In the middle ranges, however, transfers are necessary to obtain higher matching rates. A look at Figure 6.16 reveals that even in the middle ranges most riders experience zero transfers, with a few percentage experiencing 1 transfer. The maximum number of transfers ever witnessed was 4.

Figure 6.17 displays the most frequently used transfer points. This figure has been created based on the simulation results for a ridesharing system with 20,000 number of drivers, since such a system was shown to have the highest number of transfers (Figure 6.16). This figure suggests that the most important transfer points coincide with some of the red line stations, which implies good decision making in determining the station locations by Metro. This figure can also be a guide in revising the transfer points in our models in the future studies.

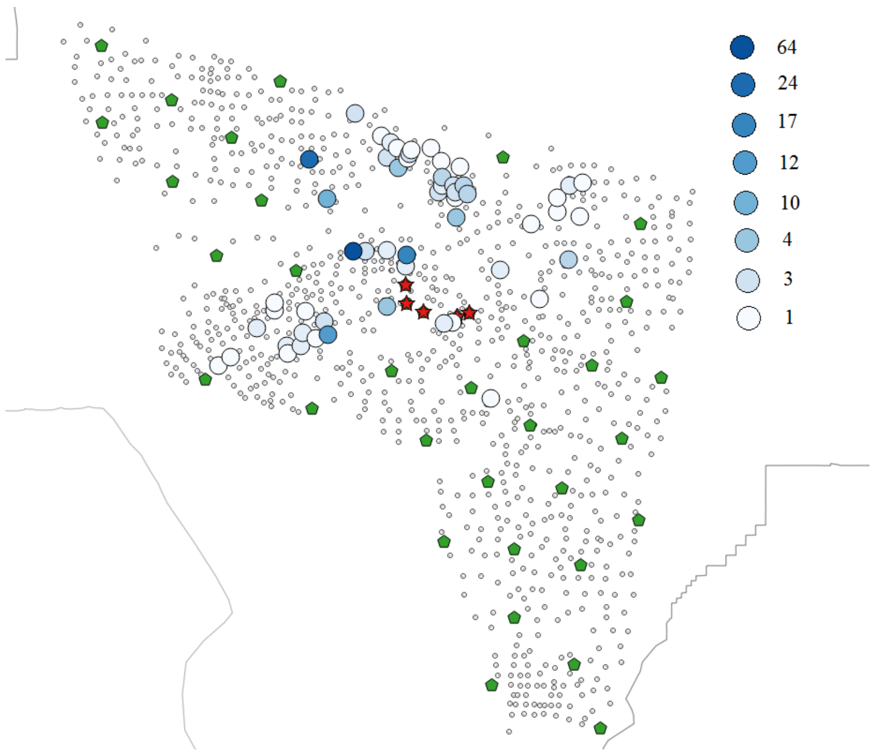


Figure 6.17: Most frequently used transfer points

Vehicle Occupancy

Figure 6.18 shows the average vehicle occupancy as a function of the supply level. This figure suggests that the average occupancy of vehicles decreases as the number of drivers in the system increases, which is intuitive, since at lower levels of supply riders are more probable to share the limited resources. The maximum vehicle occupancy, however, follows the previously observed trend of initially experiencing a rise, followed by a decline. Notice that the minimum vehicle occupancy is always higher than 2, since each matched driver carries at least one rider.

VMT Savings

Figure 6.19 displays the system-wide savings in vehicles miles traveled in the system under different levels of supply. Under lower levels of supply where the matching rate is small,

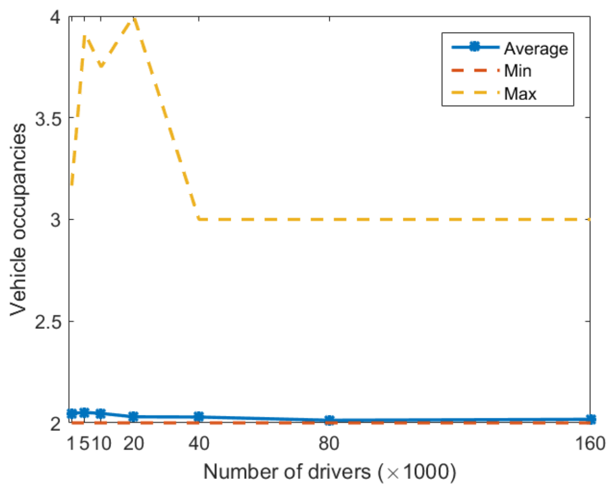


Figure 6.18: Vehicle occupancies)

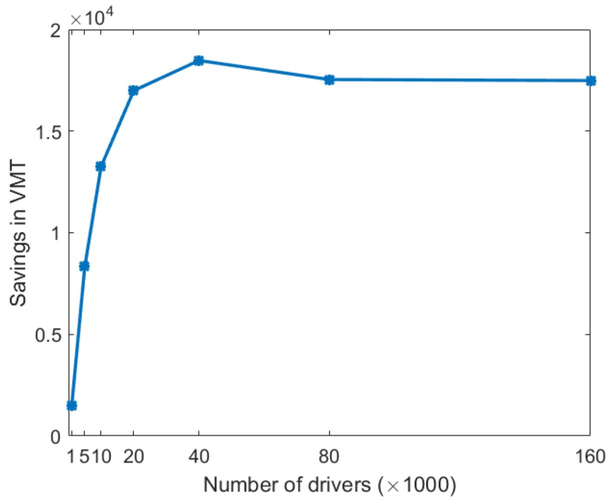


Figure 6.19: VMT savings

so is the VMT savings. VMT savings experience significant increase as the matching rate increases. However, at higher levels of supply, since drivers are abundant, vehicle occupancies drop (Figure 6.18), and consequently VMT savings experience a slight decline. In general, the analyses conducted in this section suggest that with number of drivers falling in the range between 5,000 and 20,000, the system performance reaches its peak, in terms of making the best use of resources.

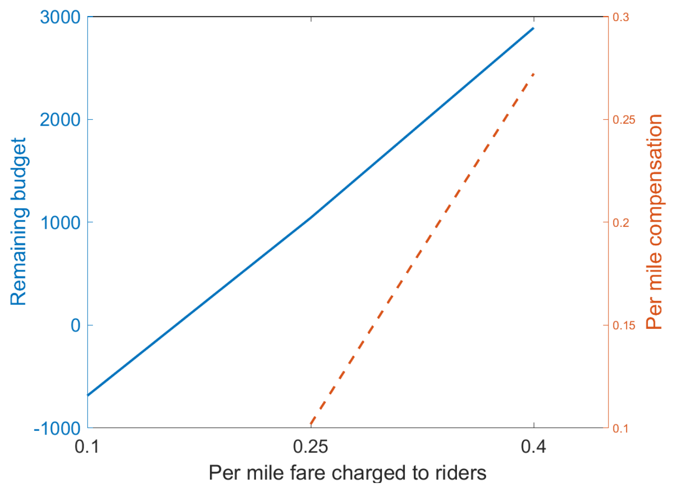


Figure 6.20: Driver Compensation

Budget Balancedness

In this section, we look into the circumstances under which the ridesharing system can be budget balanced, and operate without need for outside subsidies. We assume that the system charges a distance based fare to riders. From this total income, the system pays 0.56¢ to drivers for every additional mile they have to travel (compared to their shortest path), and then distributes the rest of the income to drivers on a per mile basis for the duration of their trips where they have been transporting passengers.

Figure 6.20 shows the remaining budget after paying drivers for their additional idle travels. This figure displays the results of three simulation runs, with 10,000 drivers, and varying per-mile fare charged to riders. Results show that when riders are charged very little (0.10¢ per mile), the system will encounter a deficit. Per-mile fares of 0.25¢ and 0.4¢ per mile, however, lead to some positive income, which can then be distributed to drivers based on their contoribution to the system. Figure 6.20 suggests that when riders are charged 0.25¢ per mile, drivers can get paid 0.1¢ per mile, and when riders are charged 0.4¢ per mile, drivers can get paid 0.27¢ per mile. Under such circumstances, the system is budget balanced and does not require subsidies.

The Rideshare-Transit Alternative

Figure 6.21 shows the percentage of riders who use the transit-rideshare option, under different values for value of time (VOT), and distance-based fares. This figure shows that, as intuition suggests, use of transit increases with the distance-based fare (under all VOT values), since the transit fare is fixed. This figure also suggests that use of transit increases with VOT, which is expected, since the LA metro red line has a high speed compared to the street network.

Although at first glance it might seem like the percentage of individuals using ridesharing as a means to connect to transit is not significant, it should be noted that the trip tables used for simulations are single-occupancy trip tables, and the individuals using the transit-rideshare option are actually increasing the current level of transit ridership. Furthermore, keep in mind that having as little as 1% of single-occupancy vehicles switch to transit would translate to a considerable increase in transit ridership. According to SCAG trip tables, 2,000,000 single occupancy vehicles travel during the morning peak hours in LA County in a working day. This adds up to a total of about 20,000 additional trips, just by the Metro red line. This number of trips distributed evenly between the roughly 50 Metro lines running during the morning peak hours indicates an additional 400 passengers in each train.

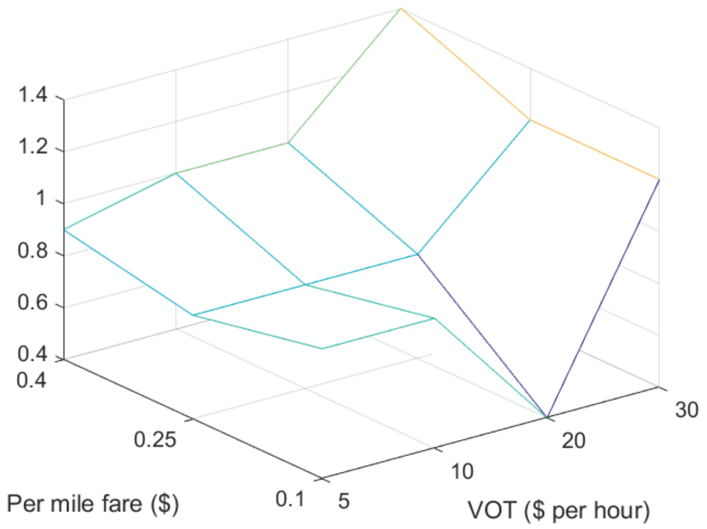


Figure 6.21: Percentage of ride requests satisfied by the transit-rideshare option)

Chapter 7

The Many-to-Many Matching Problem

In a many-to-many ride-matching problem each driver can carry multiple passengers in his/her vehicle at each moment in time, and each rider can transfer between multiple drivers. A many-to-many matching problem is computationally hard to solve, and using model (4.5) directly to solve this problem either leads to failure, or takes a long time.

In this chapter, we propose a decomposition algorithm that solves the many-to-many matching problem more efficiently, through iteratively solving multiple smaller problems, called "sub-problems". Figure 7.1 shows solution times of multiple random instances of a matching problem that differ in the number of riders and drivers. The solution times reported in this figure are obtained by directly solving model (D.1), as explained in Chapter 5 and given in Appendix D. It is interesting to notice that problem instances that include a small number of riders, but a large number of drivers (or vise versa) can be solved in a few seconds. Instances that include a large number of riders and drivers, however, are harder to solve. This observation is what motivates the forming of the sub-problems in the decomposition al-

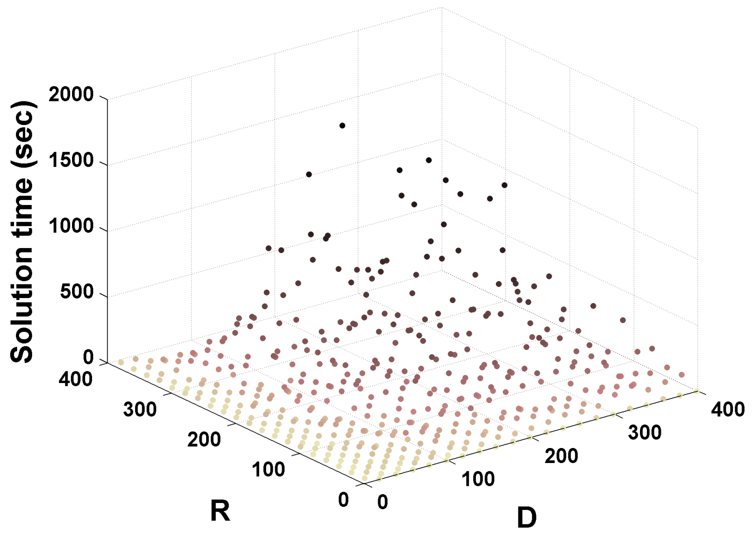


Figure 7.1: Solution time of a set of problem instances by directly solving model (D.1)(Appendix D)

gorithm. This algorithm attempts to solve the original ride-matching problem $\big($model (4.5)$\big)$, or the revised version of the problem in model (D.1), via solving a number of sub-problems that are easier to solve.

7.1 Decomposition Algorithm

The basic idea behind the decomposition algorithm is that in each iteration the algorithm solves a number of sub-problems that can represent the entire system. If the solutions to the sub-problems do not have any conflicts, the algorithm is terminated and the union of solutions to the sub-problems yields the global optimum for the original problem (proof in section 7.3.1). The algorithm flowchart is displayed in Figure 7.2.

Let R_i^k and D_i^k denote the set of riders and drivers in sub-problem k of iteration i respectively. Each sub-problem includes a subset of riders in the problem. Once the subset of riders for sub-problem k in iteration i is determined, the set of drivers can be formed as $D_i^k = \{d|\forall r \in$

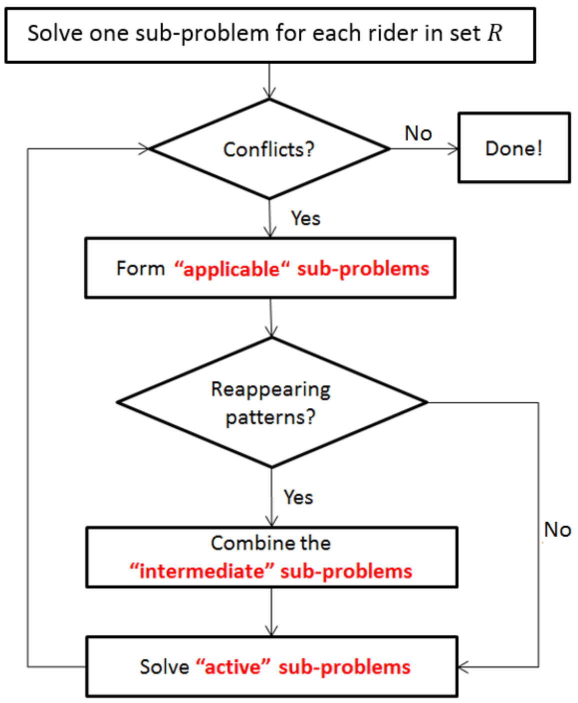

Solve one sub-problem for each rider in set R
Conflicts?
No
Done!
Yes
Form "applicable" sub-problems
Reappearing patterns?
Yes
No
Combine the "intermediate" sub-problems
Solve "active" sub-problems

Figure 7.2: The decomposition algorithm flowchart

R_i^k,$(r, d) \in M\}$.

The algorithm starts by solving $|R|$ sub-problems, each including one of the riders in set R (note that since at this initialization step each sub-problem includes a single rider, we can use the DP algorithm in Chapter 6 to solve the sub-problems in the first iteration much more efficiently). In the case of there being no conflicts between the solutions, the solution to the original problem is readily available. This happens if each rider is matched with a different driver, or if multiple riders are matched with the same driver and the driver is capable of performing all the pick-up and drop-off assignments. If not, conflicts are identified. Note that existence of conflicts between drivers' paths in different sub-problems implies that the union of solutions to the sub-problems is infeasible to the original ride-matching problem.

In each iteration, in the case of there being conflicts between solutions of the sub-problems in the previous iteration, we form the set of "applicable" sub-problems. An applicable sub-problem is comprises: (i) a group of riders from the last iteration's sub-problems with identical driver assignment (if these assignments conflict in time or space), and (*ii*) sub-problems from the previous iteration from which riders are excluded (with the remaining set of riders). Sub-problems in the previous iteration that are not applicable sub-problems will be carried out to the next iteration without any changes.

After the new set of sub-problems are formed, first we have to check to see if there are any loops between iterations. If the set of sub-problems in the current iteration is similar to the set of sub-problems in a previous iteration, the algorithm will be looping between iterations, if no measures are taken. To prevent this, we re-define sub-problems in the current iteration by forming an "intermediate" sub-problem, whenever a loop is identified. The intermediate sub-problem combines the sub-problems from the previous iteration that contribute to the loop, and hence prevent it (refer to Algorithm C in the Appendix for details).

After all the new sub-problems are determined, a decision has to be made on whether a sub-problem needs to be solved or not. Sub-problems that need to be solved are called "active" sub-problems. These sub-problems are the ones whose optimal solutions cannot be readily obtained from the previous solutions. Sub-problems that have already been solved in the previous iterations (such as non-applicable sub-problems) are not active. In addition, if a sub-problem is the union of multiple sub-problems in a previous iteration, and the solutions of these sub-problems do not conflict, the solution to the sub-problem can be readily obtained by combining the solutions of the non-conflicting sub-problems. The algorithm stops when the solutions to the current iteration's sub-problems do not have any conflicts, i.e. each driver is assigned to one route only.

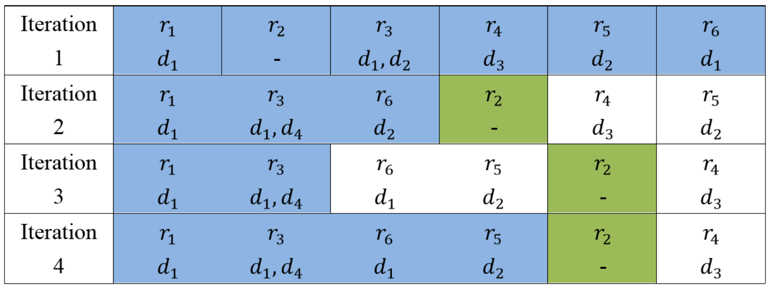
Note that using this decomposition algorithm, a large problem could be solved in the first iteration, or we might end up solving multiple sub-problems before having to solve the original problem in the last iteration. The main merit of this algorithm is that sub-problems in each iteration can be solved independently. This also indicates that parallel computing implementations are possible. The pseudo-code of the scheme is described in Algorithm C in the Appendix.

7.2 Illustrative Example

Assume that a ridesharing system has 6 riders and 4 drivers, and that all drivers have spatiotemporal proximity with all riders, i.e. $(r, d) \in M, \forall (r, d) \in R \times D$. The iterations of the decomposition algorithm are displayed in Figure 7.3. As the interest is in showing the nature of the creation of sub-problems, the actual network on which this problem is solved is left out. The active sub-problems during each iteration are displayed in blue in the figure. The problems whose solutions do not change throughout the iterations of the algorithm are displayed in green.

In iteration 1, each rider constitutes an active sub-problem. The solutions show that riders 1, 3 and 6 all have driver 1 in their solution, but in conflicting paths. Therefore, an active sub-problem of $\{r_1, r_3, r_6\}$ is formed and solved in the second iteration. Also, since rider 2 was not able to find any matches, even without facing competition from other riders, he/she will not be able to find a match in the current configuration of the system. Sub-problems $\{r_4\}$ and $\{r_5\}$ are not active sub-problems and their solutions are readily available.

The solution to the active sub-problem $\{r_1, r_3, r_6\}$ in iteration 2 indicates that the optimal matches for riders 5 and 6 are in conflict (they are both matched with driver 2, but through different paths). So they form the applicable sub-problem $\{r_5, r_6\}$. Also, since rider 6 was removed from the sub-problem $\{r_1, r_3, r_6\}$, the solution obtained for this sub-problem for riders 1 and 3 might not be optimal anymore. Therefore, a new applicable sub-problem $\{r_1, r_3\}$ is formed. However, not both of these newly formed sub-problems are active. The optimal match for rider 5 is driver 2, and the optimal match for rider 6 is driver 1. Since these two do not have any conflicts, the solution to the $\{r_5, r_6\}$ sub-problem is readily available.



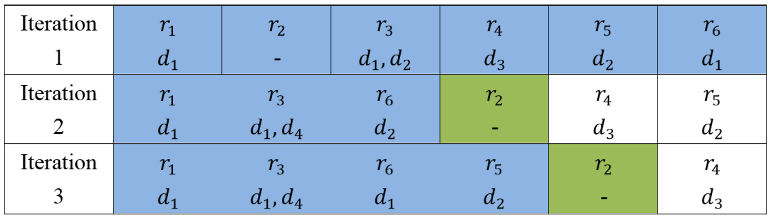
Iteration 1	r_1 d_1	r_2 -	r_3 d_1, d_2	r_4 d_3	r_5 d_2	r_6 d_1
Iteration 2	r_1 d_1	r_3 d_1, d_4	r_6 d_2	r_2 -	r_4 d_3	r_5 d_2
Iteration 3	r_1 d_1	r_3 d_1, d_4	r_6 d_1	r_5 d_2	r_2 -	r_4 d_3
Iteration 4	r_1 d_1	r_3 d_1, d_4	r_6 d_1	r_5 d_2	r_2 -	r_4 d_3

Figure 7.3: Iterations of the decomposition algorithm

The only active sub-problem in iteration 3 is $\{r_1, r_3\}$. The solution to this sub-problem suggests that two new applicable sub-problems $\{r_1, r_3, r_6\}$ and $\{r_5\}$ need to be formed, which along with two sub-problems $\{r_4\}$ and $\{r_2\}$ should constitute the set of sub-problems for iteration 4. However, we had the exact same set of sub-problems iteration 2. Therefore, in order to avoid looping, a new (intermediate) sub-problem $\{r_1, r_3, r_6, r_5\}$ is formed in iteration 4. After solving this sub-problem, we find that there are no more conflicts. So the global solution to the original problem is obtained in iteration 4. A total of 9 sub-problems needed to be solved for this solution to be obtained. However, in iteration 1, all 6 active sub-problems could be solved simultaneously.

It is possible to solve another version of the algorithm which is easier to implement, but may take longer to solve. In this simplified version, if any two riders in two sub-problems have conflicts, all the riders in the two sub-problems are combined into a new sub-problem in the following iteration. This will lead to potentially fewer iterations, but larger sub-problems to be solved in each iteration.

Let us apply this simplified algorithm to the example above. The sub-problems in each iteration are presented in Figure 7.4. Here, after solving the active sub-problem $\{r_1, r_3, r_6\}$ in iteration 2, and studying the solutions to all sub-problems, it turns out that riders 6 and 5 are both matched with driver 2, but through conflicting paths. Therefore, in the next iteration the two sub-problems $\{r_1, r_3, r_6\}$ and $\{r_5\}$ are combined. In this particular example, using the simplified version of the algorithm leads to reaching the optimal solution



Iteration 1	r_1 d_1	r_2 -	r_3 d_1, d_2	r_4 d_3	r_5 d_2	r_6 d_1
Iteration 2	r_1 d_1	r_3 d_1, d_4	r_6 d_2	r_2 -	r_4 d_3	r_5 d_2
Iteration 3	r_1 d_1	r_3 d_1, d_4	r_6 d_1	r_5 d_2	r_2 -	r_4 d_3

Figure 7.4: Iterations of the simplified decomposition algorithm

in fewer iterations, and less amount of time, since Figure 7.4 is skipping iteration 3 in Figure 7.3. Although reaching the optimal solution in fewer iterations is expected while using the simplified version of the algorithm, a smaller solution time is not a typical behavior to expect from it.

7.3 Properties of the Decomposition Algorithm

7.3.1 Optimality

During each iteration of the decomposition algorithm, we are in fact solving a relaxation of the original ride-matching problem. In the original optimization problem each driver can be assigned to a single route. By including a driver in multiple sub-problems, we might have the driver being routed differently in each. In fact, we are relaxing the constraint set that forces each driver to take a single route. Throughout the iterations, we are trying to find conflicts of this nature, and merge the sub-problems whose solutions demonstrate such a conflict, in an attempt to find a solution with no conflicts.

If the optimal solution to this relaxation satisfies the omitted set of constraints, then this solution would be optimal to the original problem as well. We have based the stopping criteria of our algorithm on this factual principle. After the final iteration of the algorithm, there exist no more conflicts between any given driver's routes in different sub-problems. Therefore, what we find through the use of the decomposition algorithm is indeed an optimal solution to the relaxation of the original problem that does not violate the set of omitted constraints. Hence the solution found is the globally optimal solution to the original problem.

It should be noted that throughout the iterations, the union of the sub-problem solutions form an infeasible solution to the original problem. It is only in the last iteration, where there are no conflicts among driver assignments, when the first feasible solution is obtained, and this feasible solution is in fact the optimal solution. Furthermore, note that although the pre-processing procedure may eliminate some of the drivers from the pool of drivers available for each rider, it does not affect the optimality of the final matching. The omitted drivers could not have contributed to the solution in any case, since they did not have spatiotemporal proximity to the riders.

7.3.2 Bounds

In this section we discuss computing upper and lower bounds on the optimal value of the objective function in model (4.5) after each iteration of the decomposition algorithm. We compute the bounds for the objective function in (10.3a), but the concept can be easily extended to any other objective function.

Clearly, if a ride-matching problem is solved to optimality, the upper and lower bounds would be similar. However, if there is a time limit on reaching a solution, we might be obliged to settle for a sub-optimal solution (as will be described later in section 7.5.4). In that case, upper and lower bounds on the optimal solution can be computed after completion of each iteration to provide insight on the quality of the solution.

The main objective in model (4.5) is to find the maximum number of served riders, and that the second term in (10.3a) has been added only for technical reasons (to ensure that the constraint sets work properly, as explained in Proposition A in the Appendix). Therefore, while solving the sub-problems, we include this term, but use only the value of the first term as the value of the objective function of the problem while computing bounds.

In section 7.3.1, we showed that the solution reached in every iteration of the decomposition algorithm is infeasible, until the last iteration. This infeasibility is caused by conflicts between itineraries of common drivers in different sub-problems. When such conflicts exist, the best case scenario is for all the riders who are receiving conflicting itineraries to be eventually served in the system. Therefore, the upper bound in each iteration can be computed as the sum of the number of served riders in all sub-problems, regardless of the conflicting driver itineraries.

It should be noted that the upper bound is strictly non-increasing with the number of iterations. The reason is that when two sub-problems have conflicting assignments during a given iteration, the riders who are receiving conflicting itineraries will be included in the same sub-problem in the following iteration. If all such conflicting riders can be served, then the upper bound to the objective function does not change. Otherwise, the upper bound will decrease.

The lower bound in each iteration can be obtained by solving a set packing problem. The optimal solution to sub-problems of a typical iteration provides us with information on the riders with conflicting itineraries, and drivers who form the itineraries of such riders. To find the tightest lower bound, we have to find the maximum number of riders who could be served under the current configuration (e.g. assuming that the driver routes are fixed). This is analogous to solving a set packing problem where the universe is the set of drivers, and the sub-sets are drivers who form each rider's itinerary.

It should be noted that unlike the upper bound that has a non-increasing trend with the number of iterations, the lower bound is not necessarily non-decreasing with iterations. In addition, note that the lower bound corresponds to a feasible solution, while the upper

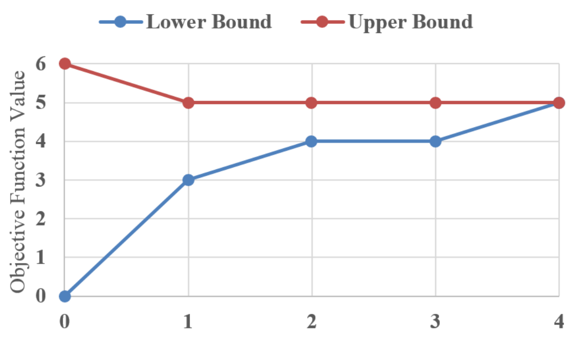


Figure 7.5: Bounds on the objective function value for the example in section 7.2

bound corresponds to a possibly infeasible solution. Figure 7.5 shows the lower and upper bounds for the example in section 7.2. Clearly the lower and upper bounds meet at the final iteration.

As we mentioned in the beginning of this section, we can follow a similar logic to compute bounds for any objective function. Since we are solving a relaxation of the original ride-matching problem at each iteration, no matter the objective function, sum of the objective functions of sub-problems always provides an upper bound on the original problem. To obtain a lower bound at each iteration, similar to what we did for the objective function in (10.3a), we have to compute a feasible global assignment from the solutions to the sub-problems at each iteration by solving a set packing problem. At each iteration, once we find the set of riders who can be served, the drivers who serve them and the itineraries of these riders and drivers are readily available, and can be used to calculate any measures that are included in the objective function to obtain a lower bound.

7.4 Numerical Study

To evaluate the performance of the proposed decomposition algorithm, we generate and solve 420 random instances of the ride-matching problem. Each problem instance has a different size in terms of number of riders and drivers, and is generated in a grid networks with 49 stations. The results reported here are averaged over the 10 runs for each problem instance.

The number of participants $|P|$ in the problem instances varies between 20 and 400, and the number of riders $|R|$ between 1 and $|P|$. Origins and destinations of trips are selected uniformly at random among the 49 stations. Earliest departure times of all trips are generated randomly within a one hour time period. The maximum ride time for a participant p is generated randomly within the window $[tt_{(OS_p,DS_p)}, 1.1\, tt_{(OS_p,DS_p)}]$. The latest arrival times are simply the sum of the earliest departure times and the maximum ride times of participants. Each vehicle is assumed to have 4 empty seats, and each rider is assumed to accept up to three transfers.

The ride-matching instances are solved on a PC with Core i7 3 GHz and 8GB of RAM. The optimization problems are coded in AMPL, and solved using CPLEX 12.6.00 with standard tuning.

In addition to problem instances that use these default parameter values, we solve additional problem instances in section 7.4.4 to conduct sensitivity analysis over some of the parameters of the problem, including the number of stations, the spatiotemporal proximity of trips, and the maximum ride time of participants. The results shown in the rest of the chapter are for solving the static versions of the problems, unless otherwise specified.

7.4.1 Pre-Processing

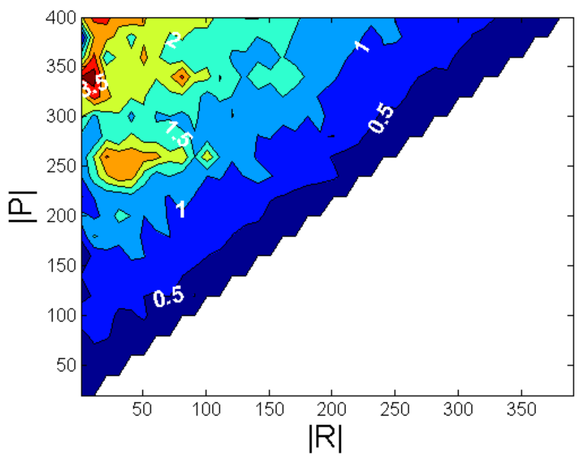
In this section, we look at the running time of the pre-processing procedure ESTAM, and examine its impact on the size of the ride-matching problem that needs to be solved. Figure 7.6a shows the contour plot of the pre-processing time as a function of the size of the problem. The pre-processing time in this chapter is the sum of two components: the time required to generate the set of feasible links for each participant, and the time required to find the set of feasible drivers for each rider.

The former component consists of the time spent on generating the reduced networks, and forming sets of feasible links for participants using forward and backward movements. This task can be completed when a participant registers a request in the system, and hence is not a bottleneck when it comes to solving the ride-matching problem in real-time in a rolling time-horizon implementation. Even if a large number of participants join the system at the same time, the computations can be done independently, and implemented in a parallel computing system.

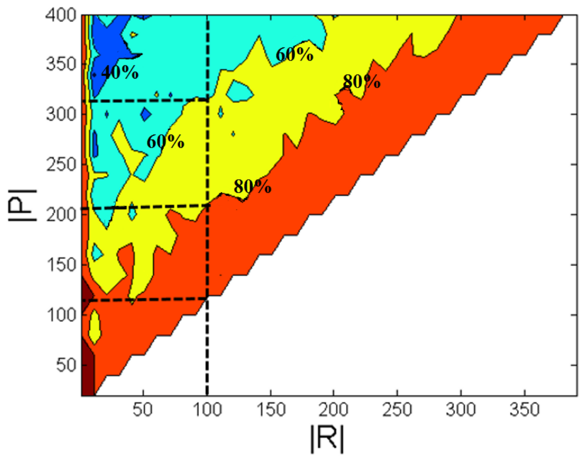
The latter component of the pre-processing time consists of the time required to compare each rider's set of feasible links to those of the drivers. Similar to the first component, these computations can be done once a rider joins the system (with all registered drivers), and updated continuously as drivers keep joining the system.

The total pre-processing time depends on the size of the problem. For the 420 problem instances solved in this section, the maximum time was 3.5 seconds. The distribution of the pre-processing times over the randomly generated instances are displayed in Figure 7.6a. The pre-processing procedure managed to reduce the average size of the link sets for participants to 0.01% of the size of the original link set L.

Not only does the pre-processing procedure lead to indirect savings in solution time by limiting the size of the input sets to the optimization problem, but more importantly a quick scan of the participants' feasible links can lead to direct and more considerable savings in solution time. For a rider to be served, he/she should have spatiotemporal proximity with some driver at both origin and destination stations. Whether such proximity exists can be easily examined using the sets of feasible links. Riders who do not have spatiotemporal proximity at both origin and destination stations can be filtered out from the system. Figure 7.6b shows the fraction of riders filtered out from each problem instance during the pre-



(a) Pre-processing time (sec)



(b) Percentage of filtered riders

Figure 7.6: The pre-processing procedure

processing procedure. This figure suggests that, for a given number of riders, the lower the number of drivers, the higher the number of filtered riders, as expected.

7.4.2 Value of a Multi-hop Solution

A ridesharing system can use a variety of approaches to match riders with drivers. Ride-matching methods could differ in multiple aspects, ranging from the flexibility in routing of drivers, to the type of routes devised for riders (single- or multi-hop). Matching methods can be formulated as optimization problems, and solved to optimality. However, different ride-matching methods lead to different system performance levels.

In this section, in order to investigate the effectiveness of the ridesharing system defined in this study, we compare its performance with a few more common ride-matching methods. Figure 7.7 compares the cumulative number of served riders in the 420 randomly-generated problem instances using five different matching methods. The problem instances in this figure are sorted by the number of participants.

The first matching method labeled as the "OD-based" in Figure 7.7 matches riders and drivers who share the same origin and destination locations, if their time windows allow. The second matching method, labeled as “Single-hop, Fixed-route” matches each rider with one driver only, where the drivers’ routes are pre-specified and fixed (although they still have flexibility in departure time). The third matching method labeled as “Multi-hop, Fixed-route” allows riders to transfer between drivers. The drivers’ routes, however, remain pre-specified. The fourth matching method labeled as “Single-hop, Flexible-route” matches each rider with one driver only. In this method, however, the ridesharing system takes over routing of the drivers. The fifth and final matching method labeled as “Multi-hop, Flexible-route” allows riders to transfer between drivers. The driver routes are not pre-specified, and will be determined by the system.

As Figure 7.7 indicates, the OD-based matching method provides the least number of matches. The Single-hop, Fixed-route method does significantly better, since in this method not only can drivers carry the riders who share the same origin and destination locations with them (as in the OD-based method), but also they can carry passengers whose origin and destination locations lie on their pre-specified routes. The Multi-hop, Fixed-route matching method produces results that are slightly superior to the Single-hop, Fixed-route method, implying that allowing riders to transfer between drivers is in general beneficial to the system.

The Single-hop, Flexible-route and Multi-hop, Flexible-route methods lead to considerable improvements in the number of matches. This implies that having the system route the drivers can be one of the most influential features of a ridesharing system. Comparison of the Single-hop, Flexible-route and Multi-hop, Flexible-route methods suggests that allowing transfers in the system is the second most important factor, after routing drivers. In addition, comparing the Single-hop, Flexible-route and Multi-hop, Flexible-route methods with Single-hop, Fixed-route and Multi-hop, Fixed-route methods suggests that the value of a multi-hop solution becomes more prominent when the driver routes are not pre-specified.

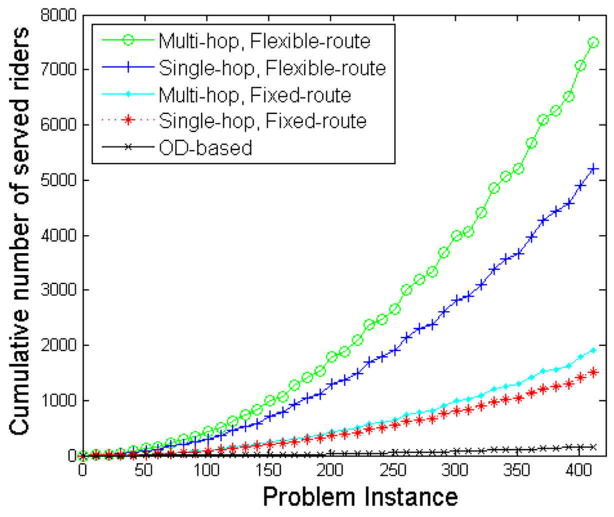


Figure 7.7: Cumulative number of served riders using five different matching algorithms

Figure 7.7 measures the efficiency of the matching methods based on the number of served riders. What this figure fails to demonstrate is how measures of quality of service, such as number of transfers and waiting times in transfer stations for riders, and the average vehicle occupancy and the extra time spent in the network by drivers compare between these methods. Table 7.1 displays these measures of quality of service along with some additional metrics to assess the five matching methods from different perspectives.

The results presented in this table are averaged over 10 runs with 400 participants, 200 of which are drivers and 200 are riders. This problem composition is selected because it yields the highest number of served riders, regardless of the matching method. The earliest departure times of all trips are randomly generated within a 60 minute time period, and the trip origin and destination stations are selected randomly among the 49 stations. The quality of service measures in this table are averaged over all participants in all the randomly-generated problems.

Table 7.1 suggests that in addition to serving higher number of riders, the multi-hop matching methods also engage higher number of drivers, compared to their single-hop counterparts. Although the ultimate purpose of a ridesharing system is serving riders, involving higher number of drivers is important as well, because users who do not get matched might stop registering in the system after a few failed attempts. Another interesting point is that in multi-hop matching methods the number of matched drivers is higher than number of served riders. In spite of this observation, the average number of riders a driver in a multi-hop system carries is slightly higher than the average number of riders carried by a driver in a single-hop system.

Another interesting observation is that vehicle occupancies are highest in a multi-hop fixed-route system. In such a system, because driver's routes are fixed, a smaller percentage of drivers get matched. However, drivers who get matched are the ones who drive on "popular" routes, and therefore end up with higher occupancy rates. In other words, although the system is not routing drivers, those drivers who try to optimally route themselves will benefit more compared to the case where the system routes all drivers.

Table 7.1 suggests that the measures of quality of service stay within reasonable ranges for all matching methods. It should be noted, however, that these measures correlate with the problem parameters in general, and are more sensitive to some parameters of the problem than others, as we will see in section 7.4.4.

Table 7.1: Quality of service measures for the five matching methods in a system with relatively low spatiotemporal proximity among trips

	Riders			Drivers		
Matching method	Num. (%) served	Min\avg.\max num. transfers	Min\avg.\max wait in transfer (min)	Num. involved	Min\avg.\max extra travel time (min)	Min\avg.\max riders on board
OD-based	5 (3%)	NA	NA	5 (3%)	NA	1.0\1.00\1.0
Single-hop, Fixed-route	11 (6%)	NA	NA	8 (4%)	NA	1.0\1.37\3.1
Multi-hop, Fixed-route	16 (8%)	0.0\0.63\1.9	0.0\0.38\2.5	18 (9%)	NA	1.0\1.44\3.2
Single-hop, Flexible-route	32 (16%)	NA	NA	26(13%)	0.0\3.04\8.1	1.0\1.23\2.3
Multi-hop, Flexible-route	52 (26%)	0.0\0.50\2.6	0.0\0.63\2.1	62(31%)	0.0\2.13\6.9	1.0\1.26\1.8

Table 7.2: Quality of service measures for the five matching methods in a system with relatively high spatiotemporal proximity among trips

	Riders			Drivers		
Matching method	Num. (%) served	Min\avg.\max num. transfers	Min\avg.\max wait in transfer (min)	Num. involved	Min\avg.\max extra travel time (min)	Min\avg.\max riders on board
OD-based	21(11%)	NA	NA	19(10%)	NA	1.0\1.11\2.4
Single-hop, Fixed-route	79(40%)	NA	NA	62(31%)	NA	1.0\1.27\4.2
Multi-hop, Fixed-route	89(45%)	0.0\1.51\2.5	0.0\0.91\3	104(52%)	NA	1.0\1.91\3.1
Single-hop, Flexible-route	104(52%)	NA	NA	79(35%)	0.0\2.47\6.1	1.0\1.32\2.9
Multi-hop, Flexible-route	152(76%)	0.0\2.22\2.9	0.0\1.28\5	181(86%)	0.0\4.75\8.8	1.0\1.26\3.6

The problem instances whose performance we studied in Table 7.1 represent a ridesharing system that is spatiotemporally sparse. To compare the matching methods in a more realistic setting, we generated 10 random instances of a ridesharing system with higher spatiotemporal proximity among trips. Similar to the problem instance studied in Table 7.1, this problem instance contains 400 participants, 200 of whom are riders and 200 are drivers. The earliest departure times of all trips is generated within a 30 minute time period. The network is clustered into two sets of non-intersecting sections, where all trip origins are randomly selected from stations located in one of the sections, and all trip destinations from the other. This trip generation procedure significantly increases the spatiotemporal proximity among trips. The number of served riders and matched drivers using all five ride-matching methods along with some measures of quality of service are displayed in Table 7.2. This table suggests that all matching methods serve higher number of riders in a more spatiotemporally-dense ridesharing system.

7.4.3 Percentage of Satisfied Ride Requests

Figure 7.8 displays the contour plots of the percentage of satisfied ride requests for the randomly generated problem instances. As intuition suggests, for a given number of riders, the percentage of served requests increases with the number of drivers in the system. Analyzing this type of graph can be especially helpful when one is interested in the critical mass of participants required to keep the system up and running, or when looking to set marketing strategies. For example, if we have 100 registered riders in a network where the origin and destination of trips are completely random, and aim to serve at least 40% of the riders, then we know we need at least 230 drivers (330 participants), and can tailor our marketing strategies accordingly.

Note that the percentages shown in Figure 7.8 are the result of uniformly distributed OD

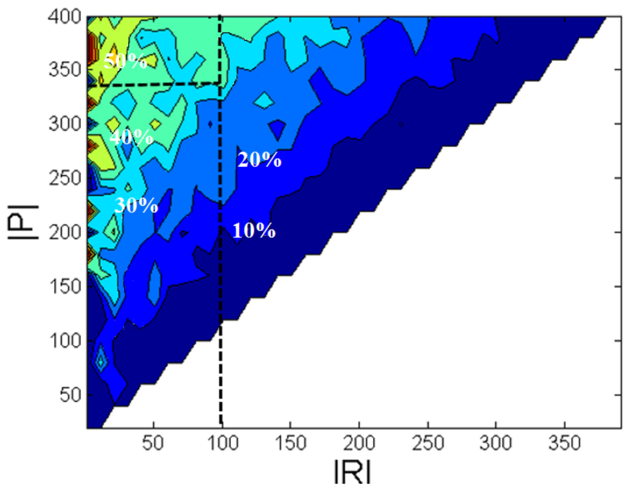


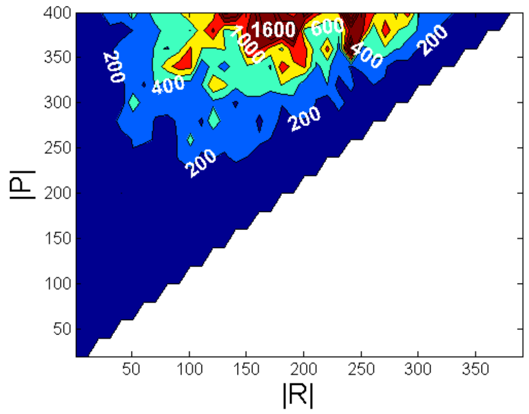
Figure 7.8: Percentage of riders served

patterns in the study problems, which is a worst case scenario with a much less chance of joint rides than found in real networks, where trip distributions cause high demands for certain OD pairs, hence increasing the spatial proximity of trips in the network. It is expected that real networks can operate with much better ratios between drivers and riders. As a consequence, marketing attempts in real networks can be targeted towards the more popular OD pairs.

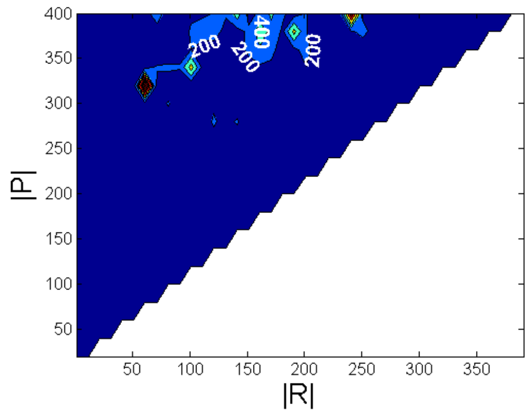
7.4.4 Algorithm Performance

Figures 7.9a and 7.9b compare the time required to solve the randomly generated ride-matching problems to optimality, without and with the decomposition algorithm. Solution times in Figure 7.9a were obtained by solving model (D.1) directly, and solution times in Figure 7.9b were obtained by solving the same model using the decomposition algorithm. Comparison of these two figures suggests considerable savings in solution times can be obtained using the decomposition algorithm, without any trade-offs in terms of solution accuracy.

Figure 7.10 shows the number of iterations of the decomposition algorithm it takes to solve the problems to optimality. Note that the higher numbers of iterations correspond to prob-



(a) Original matching problem solution times (sec)



(b) Decomposition algorithm solution times (sec)

Figure 7.9: Improvements in solution times. Contour plots of solution times in seconds.

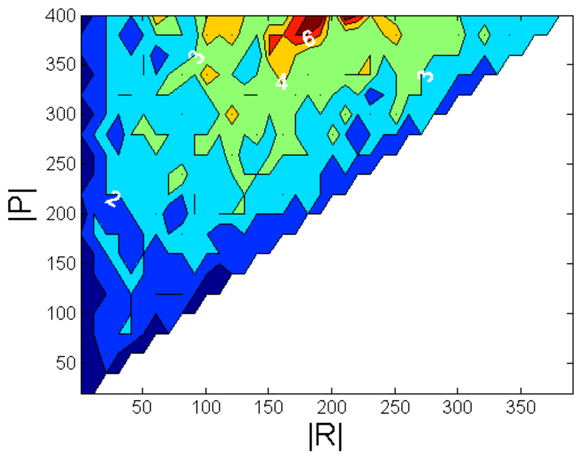


Figure 7.10: Number of decomposition algorithm iterations

lems with higher solution times. None of the problem instances requires more than 7 iterations to be solved to optimality using the decomposition algorithm.

7.4.5 Transfers

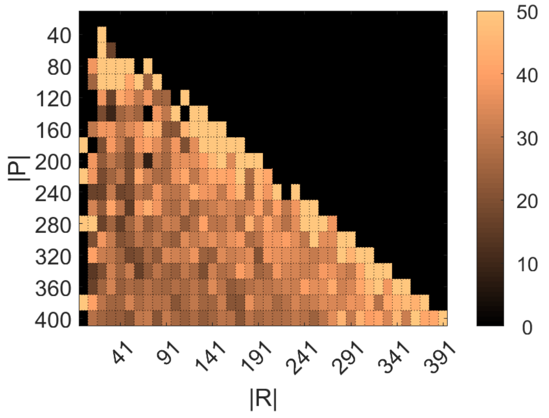
Transfers usually connote discomfort, and understandably so; since public transportation commonly runs under a fixed schedule, making transfers could lead to high wait times and large deviation from one's shortest route. Note, however, that ride-sharing systems of the future could be fundamentally different in nature, and better user-acceptance of transfers is possible depending on the ease of transfers and reduction in transfer times offered by such systems. This will naturally depend on the market penetration of the system as well as user-side behavioral changes. While this remains speculative, in this study we allow riders to specify their maximum number of transfers, and incorporate the maximum ride times requested by riders and drivers in our models. These input parameters could vary among individuals depending on their accessibility to personal vehicles, and value of time. As the individual riders' own transfer preferences are incorporated, this will help ease concerns on the perceived quality of service being affected by transfers.

Figure 7.11 shows the distribution of number of transfers in the 420 problem instances we solved in this section. Even though in our numerical experiments we assume that riders are comfortable with up to 3 transfers, this figure suggests that the majority of trips have zero or one transfer.

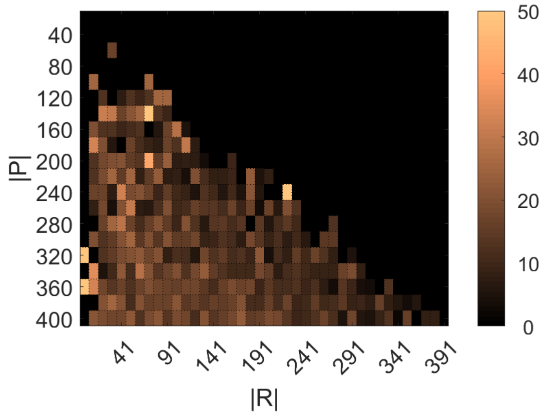
A further reason to include multiple transfers is to combine ride-sharing systems to other modes, as we saw in Chapter 6. It is straight-forward to include, say, a fixed transit system simply as high-capacity drivers with fixed routes in the formulation. In such cases, more than one-transfer between another mode and the ride-share mode (near the origin and destination, for example) is reasonable, and this also highlights the need for not limiting the number of transfers.

7.4.6 Sensitivity Analysis

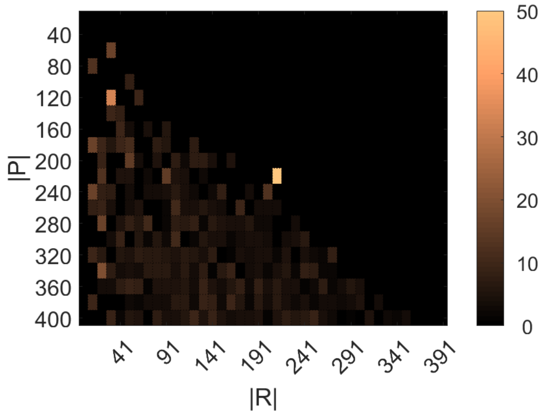
In order to perform sensitivity analysis over some of the parameters of the ride-matching problem, we generate and solve two additional series of matching problems. All these problems contain 400 participants, with equal number of riders and drivers. This configuration is selected because Figures 7.8 and 7.9 suggest that problems with equal number of riders



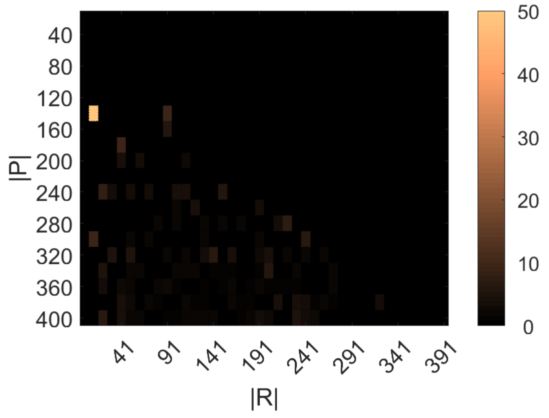
(a) Percentage of served riders with zero transfers



(b) Percentage of served riders with one transfer



(c) Percentage of served riders with two transfers



(d) Percentage of served riders with three transfers

Figure 7.11: Distribution of transfers

and drivers produce the highest number of matches, and require the most computational resources. For the following problems, we maximize an objective function that includes two main terms: sum of the total number of served riders ($\sum_{r \in R} z_r$), and the weighted total distance traveled by riders ($-\sum_{r \in R} W_r \sum_{d \in D:(r,d) \in M} \sum_{\ell=(t_i,s_i,t_j,s_j) \in L_{rd}} (t_j - t_i) y_\ell^{rd}$). The term $W_r = 1/(T_r^{TB} + 1)$ (where T_r^{TB} is the maximum ride time for rider r) is considered as the weight for the travel time of rider r in this objective function to ensure that maximizing the total number of served riders remains the main priority of the system.

Table 7.3 shows the performance of the decomposition algorithm in solving 5 problem instances in a network with 49 stations. The field "release period" in this table shows the period of time during which the earliest departure times of participants is generated. For this field, we use two values of 30 and 60 minutes. Problem instances with release period of 30 minutes incorporate higher temporal proximity between trips.

The field "Dir. trips" in Table 7.3 has a binary value. Value of zero for this field is used when the trip origin and destination locations are generated randomly in the network. In problem instances with value of 1 for this field, we use a clustered network. In a clustered network, trips experience a higher degree of spatial proximity.

Finally, we use two values of 1.1 and 1.2 for the participants' maximum ride times. Value of 1.1/1.2 for this parameter indicates that all participants are assumed to have a maximum ride time that is 1.1/1.2 times their shortest path travel time.

The field "Optimal" in Table 7.3 indicates whether a problem has been solved to optimality (Y), or we ran out of memory when solving at least one of the sub-problems (the original problem in case of a static problem), and had to report the solution obtained by the heuristic described in section 7.5.4.

All these problem instances have been solved as both static problems, and dynamic problems with different re-optimization periods. For each problem instance, "NA" under the field

"Re-optimization period" indicates that the problem is solved in a static setting, assuming all participants have registered their trips before the matching problem is solved. Other values under this field show the re-optimization period. We assume that the demand for re-optimization period i arrives within the window $[(i-1)k, ik]$, where k is the length of the re-optimization period. In addition, after each re-optimization period, we fix the itineraries of all matched riders and drivers, and include the fixed itineraries of matched drivers in the problems solved for the following re-optimization periods, if their travel time windows allow.

Not surprisingly, static versions of all five problem instances have the largest number of matches and the largest solution times compared to their dynamic counterparts, since static problems include all participants. Following the same logic, given a fixed number of participants, longer re-optimization periods translate to a larger pool of participants in the matching problems, and higher matching rates.

For each problem, we provide statistics on the number of rider transfers (based on the itineraries of matched riders), and the waiting time of riders during transfers (based on the itineraries of riders who experience transfers). As a general trend, all these values decrease with the length of the re-optimization period, although the average waiting time of riders during transfers is in general negligible, especially given the potential uncertainties in travel times.

Table 7.3 also reports the number of matched drivers, and the statistics on the extra travel time they have to spend in the network (compared to their shortest path travel times). This table suggests that number of matched drivers decreases with the length of the re-optimization period. However, no monotonic relationship can be observed between the drivers' extra time spent in the network, and the length of the re-optimization period.

Compared to the first problem instance, problem instances 4, 5, and 6 incorporate higher temporal, spatial, and spatiotemporal proximity among trips, respectively. Although all these problem instances lead to higher number of served riders compared to the first problem instance, it is interesting to note that the effect of higher spatial proximity on the number of served riders is more significant than the effect of higher temporal proximity. An interesting point to notice is that at 5-min re-optimization periods, all the problem instances can be solved in less than 1 minute.

The highest number of served riders is obtained in problem instance 2, where the participants' time budgets are the highest. However, notice that the measures of quality of service, namely the average extra travel time for drivers, and the average number of transfers and transfer wait time for riders, are also considerably higher compared to other problem instances. The matching rate and quality of service measures reported on problem instance 3 in which riders are (naturally) assumed to have a higher maximum ride time than drivers lie between those of instances 1 and 2.

Table 7.4 displays problem instances similar to the ones in Table 7.3 in terms of problem parameters, but in a network with 100 stations. We should point out that we could not solve the static version of problem instance 5 to optimality (ran out of memory), and instead used the heuristic approach described in section 7.5.4. The two numbers reported under the "Num. served" field show the lower and upper bounds on the optimal solution (the lower bound solution is the feasible solution that can be eventually used). Note that if one encounters such a problem due to lack of computational capacity, they could opt for re-optimizing the system more periodically.

In general, Table 7.4 demonstrates the same trends as in Table 7.3. The most prominent difference that can be observed comparing the two tables is that Table 7.4 has fewer number of served riders. This is an expected result, since by increasing the number of stations and keeping the same number of participants, we are in fact decreasing the spatial proximity among trips. In practice, however, increased number of stations could lead to potentially higher levels of demand, as higher number of stations translates into higher accessibility to the ridesharing system.

In the following section, we present methods to make the solution algorithm more appropriate for dynamic implementations by reducing the solution times even further.

Table 7.3: Sensitivity study over the problem parameters. All instances are generated with 400 participants composing of 200 riders and 200 drivers in a randomly generated network with 49 stations

					Riders			Drivers		Solution	
Problem Instance	Dir. trips	Release period	Time budget rider/driver	Re-optimization period (min)	Num. served	Min\avg.\max num. transfers	Min\avg.\max wait in transfer (min)	Num. Involved	Min\avg.\max extra time (min)	time (sec)	Optimal *Y/LB*
1	0	60	1.1/1.1	NA	69	0.0\0.88\3.0	0.0\0.58\3.0	98	1.0\2.67\3.0	518	*Y*
1	0	60	1.1/1.1	10	27	0.2\0.47\1.0	0.0\0.31\1.8	44	1.8\2.96\3.4	16	*Y*
1	0	60	1.1/1.1	5	20	0.4\0.45\0.8	0.0\0.36\1.2	36	1.1\2.17\3.2	6	*Y*
2	0	60	1.2/1.2	NA	150	0.0\2.09\2.1	0.0\2.50\4.1	144	0.0\11.17\15	1920	*Y*
2	0	60	1.2/1.2	10	72	0.0\0.56\1.7	0.0\1.28\3.1	105	2.9\8.50\11	174	*Y*
2	0	60	1.2/1.2	5	67	0.0\0.36\1.7	0.0\0.43\3.3	87	3.6\6.50\9.2	29	*Y*
3	0	60	1.2/1.1	NA	112	0.0\0.70\2.4	0.0\1.5\3.1	123	0.1\3.10\5.4	718	*Y*
3	0	60	1.2/1.1	10	43	0.0\0.45\1.2	0.0\0.81\3.0	80	2.0\3.05\5.1	53	*Y*
3	0	60	1.2/1.1	5	38	0.0\0.42\0.9	0.0\0.43\2.1	67	2.6\4.20\4.2	11	*Y*
4	0	30	1.1/1.1	NA	87	0.0\0.90\30	0.0\0.74\1.5	115	0.75\2.32\3.3	2357	*Y*
4	0	30	1.1/1.1	10	54	0.0\0.73\2.3	0.0\0.59\3.3	80	1.7\3.14\4.2	201	*Y*
4	0	30	1.1/1.1	5	48	0.0\0.45\1.5	0.0\0.48\3.7	68	0.5\2.55\4.3	12	*Y*
5	1	60	1.1/1.1	NA	137	0.0\2.10\3.0	0.0\1.67\7.0	158	0.0\2.40\8.0	1104	*Y*
5	1	60	1.1/1.1	10	73	0.0\0.83\2.3	0.0\0.59\2.8	107	0.0\2.45\7.06	52	*Y*
5	1	60	1.1/1.1	5	57	0.0\0.65\1.7	0.0\0.63\1.8	85	0.3\3.00\3.4	22	*Y*
6	1	30	1.1/1.1	NA	152	0.0\2.22\3.0	0.0\1.28\8.0	192	0.0\2.34\8.0	2750	*Y*
6	1	30	1.1/1.1	10	116	0.0\0.82\2.6	0.0\0.48\3.7	133	0.0\2.55\8.0	142	*Y*
6	1	30	1.1/1.1	5	101	0.0\0.65\1.5	0.0\0.43\2.2	112	0.1\1.95\6.2	48	*Y*

Table 7.4: Sensitivity study over the problem parameters. All instances are generated with 400 participants composing of 200 riders and 200 drivers in a randomly generated network with 100 stations

					Riders			Drivers		Solution	
Problem Instance	Dir. trips	Release period	Time budget rider/driver	Re-optimization period (min)	Num. served	Min\avg.\max num. transfers	Min\avg.\max wait in transfer (min)	Num. Involved	Min\avg.\max extra time (min)	time (sec)	Optimal *Y*/*LB*
1	0	60	1.1/1.1	NA	61	0.0\1.81\2.7	0.0\1.80\10	94	0.0\4.40\7.0	501	*Y*
1	0	60	1.1/1.1	10	18	0.5\1.07\1.8	0.2\1.15\3.7	39	0.5\5.86\9.1	40	*Y*
1	0	60	1.1/1.1	5	11	0.3\0.48\0.7	0.1\0.40\0.8	20	0.3\3.86\8.2	36	*Y*
2	0	60	1.2/1.2	NA	138	0.0\2.38\2.5	0.3\2.36\6.2	124	0.0\16.18\20.2	1339	*Y*
2	0	60	1.2/1.2	10	59	0.1\1.65\2.1	0.0\1.05\5.1	83	1.0\14.30\19.8	515	*Y*
2	0	60	1.2/1.2	5	39	0.0\0.43\0.9	0.1\0.89\3.9	63	2.1\12.22\18.2	316	*Y*
2	0	60	1.2/1.2	2	32	0.0\0.35\0.6	0.0\0.70\3.1	40	0.1\8.06\14.0	115	*Y*
3	0	60	1.1/1.2	NA	85	0.1\1.82\2.5	0.3\2.11\4.1	102	0.0\2.13\6.2	1020	*Y*
3	0	60	1.1/1.2	10	35	0.3\1.50\1.9	0.1\1.19\4.0	49	0.2\3.91\7.1	180	*Y*
3	0	60	1.1/1.2	5	21	0.2\0.45\1.1	0.1\0.78\3.8	30	0.1\3.87\7.3	115	*Y*
4	0	30	1.1/1.1	2	36	0.2\1.0\1.5	0.2\0.80\2.0	63	0.1\4.46\8.2	866	*Y*
4	0	30	1.1/1.1	10	23	0.1\0.85\1.7	0.3\1.74\3.4	42	0.1\4.06\9.1	339	*Y*
4	0	30	1.1/1.1	5	15	0.0\0.44\0.7	0.3\0.67\2.3	23	0.1\3.23\7.6	31	*Y*
5	1	60	1.1/1.1	NA	132	0.5\2.42\2.9	0.4\1.86\4.1	145	0.0\6.68\8.9	856	*Y*
5	1	60	1.1/1.1	10	29	0.1\0.85\1.2	0.0\0.69\2.1	49	0.1\4.02\7.1	124	*Y*
5	1	60	1.1/1.1	5	19	0.1\0.74\0.9	0.1\0.40\1.9	32	0.2\4.50\7.1	30	*Y*
6	1	30	1.1/1.1	NA	[78,128]	0.4\2.42\2.7	0.1\1.27\3.8	161	0.2\6.16\10.1	753	*LB*
6	1	30	1.1/1.1	10	61	0.1\0.97\1.2	0.1\0.97\2.1	99	0.3\4.90\6.9	2140	*Y*
6	1	30	1.1/1.1	5	40	0.1\0.78\1.2	0.1\0.82\1.9	64	0.1\2.69\4.5	447	*Y*
6	1	30	1.1/1.1	2	28	0.0\0.50\0.7	0.1\0.69\1.5	42	0.0\1.05\3.1	123	*Y*

7.5 Application in Practice

The scalability of the solution algorithm to solve the ride-matching problem may become an issue in practice. Larger areas of coverage call for larger number of stations and larger number of participants. Larger ride-matching problems can lead to higher solution times and reduce the effectiveness of the proposed algorithm for dynamic applications.

Replacing the original network with reduced networks of participants can address this problem to a large extent. Even though a ridesharing company could cover a very large area, the distance traveled by participants is usually within a limited (but not necessarily small) circumference of their homes, e.g. within city limits, and therefore their reduced networks contain a limited number of stations. It is only in rarer cases that people might require to, or be willing to, travel larger distances in a dynamic rideshare system. Thus, it may be expected that larger reduced networks would be rarer.

In order for the ridesharing system to cover longer (such as inter-city) trips, the system can locate one or two important stations in each city, and use only those set of stations for riders with long-distance trips. In addition, drivers who are selected to be included in the ride-matching problems with such riders can be limited to those who are making inter-city trips themselves. Using drivers with shorter trip lengths in such settings would not be practical due to the large number of transfers that forming an itinerary with short-distance drivers would require.

In the interest of lower solution times, in this section we suggest three different approaches to simplify the problem and obtain high quality heuristic solutions to the ride-matching problem within a reasonable time period.

7.5.1 Re-optimization Period

The solution times of the numerical experiments in the previous section (presented in Figure 7.9) are for a system whose participants have all registered their trips ahead of time, say, prior to the onset of a morning rush hour (static ridesharing). However, in real systems, it is not always the case that the demand is known in advance, and hence the system needs to be re-optimized periodically to take into consideration requests that arrive in real-time. Consequently, the problem instances that need to be solved will be smaller in size.

Figure 7.9 can be used to get an estimate on the solution times of matching problems for various re-optimization periods. For example, envision a system with 400 participants in a morning rush hour, half of whom have registered before the start of the rush hour, with the rest arriving uniformly during a one-hour period. In a rolling time-horizon implementation where the ride-matching problem is solved every five minutes, the size of the problem instance that needs to be solved at each period hardly ever goes over 200 participants (this size decreases as we proceed through the time horizon), and so the solution times will be limited to one minute. This implies that in a system with 400 participants, we can even solve the ride-matching problem once every minute, and respond to the real-time demand more effectively. It is easy to see the trade-off between the matching rate (which increases with the size of the re-optimization period) and the solution times.

7.5.2 Restricting the Number of Transfer Stations

In the ride-matching problem in model (4.5), itineraries are devised assuming transfers can occur at any station. In practice, we can identify stations where transfers are more likely to occur, and limit the set of transfer stations. Let us denote by S_T the set of stations where transfers are allowed. The flow conservation constraint (10.3g) should then be replaced by two sets of constraints, (7.1) and (7.2). Constraint set (7.1) and (7.2) both model flow conservation. Constraint set (7.1) covers stations where transfers are allowed, and constraint set (7.2) covers the rest of the stations.

$$\sum_{d \in D'} \sum_{\substack{t_i, s_i: \\ \ell=(t_i,s_i,t,s) \in L}} y_\ell^{rd} = \sum_{d \in D'} \sum_{\substack{t_j, s_j: \\ \ell=(t,s,t_j,s_j) \in L}} y_\ell^{rd}; \; \forall r \in R, \forall t \in T_r, \forall s \in S_T \backslash \{OS_r \cup DS_r\} \tag{7.1}$$

$$\sum_{\substack{t_i, s_i: \\ \ell=(t_i,s_i,t,s) \in L}} y_\ell^{rd} = \sum_{\substack{t_j, s_j: \\ \ell=(t,s,t_j,s_j) \in L}} y_\ell^{rd}; \; \forall r \in R, \forall d \in D, \forall t \in T_r, \forall s \in S \backslash \{S_T \cup OS_r \cup DS_r\} \tag{7.2}$$

To study the impact of limiting transfers to certain stations on the quality of the solutions and solution times, we experimented with the 420 randomly generated problem instances. For each problem instance, we studied the solution to model (D.1) where all stations are transfer stations, and identified the top 80% of stations in terms of frequency of transfers. We then limited transfers to those stations only. Figure 7.12 shows the ratio of reduction in solution times (new solution times divided by the old solution times). The reduction in computation time can range from 0 to 60%. It is interesting to observe that the higher savings in solution times are obtained for problem instances with higher original solution times (see Figure 7.9a).

In terms of solution quality, the number of served riders using the top 80% transfer stations remained on average within 95% of the number of served riders in the original solution, suggesting that the trade-off between the solution time and quality in dynamic settings may be in favor of restricting the number of transfer stations. These results are similar to the results found by Masson et al. (2014b). In their study of a multi-modal system designed to carry goods using a combination of excess bus capacities and city freighters, they found that 60% of the transfers took place in two bus stops that were closer to a large body of customers.

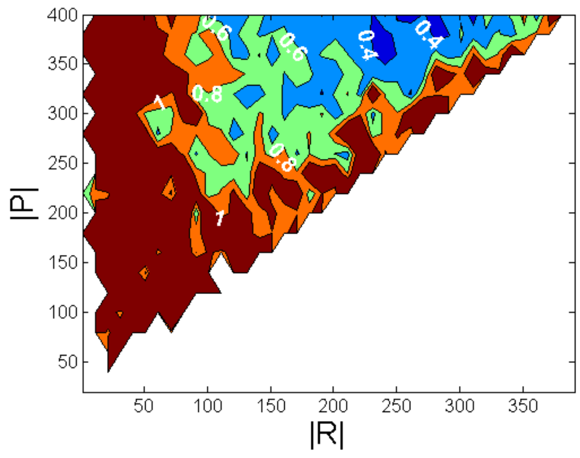


Figure 7.12: Ratio of new to old solution times after limiting the number of transfer stations

7.5.3 Selecting a Time Interval

In the numerical experiments in this chapter, we discretized the study time horizon into 1-minute time intervals. In this section, we study the impact of using larger time intervals on the solution times, and discuss when using larger time intervals is appropriate.

The impact of using larger time intervals on solution times is not obvious without running experiments. On one hand, using larger time intervals will lead to smaller link sets produced by the pre-processing procedure. On the other hand, using larger time intervals can cause higher degrees of spatiotemporal proximity among trips, which can lead to higher solution times.

Figure 7.13 shows the solution times for the 420 randomly generated problems, solved using the decomposition algorithm with 5-minute time intervals. This figure suggests that utilizing larger time intervals leads to lower solution times. It should be noted that as the participants' maximum ride times increase, so do the size of the link sets, and therefore the savings in solution times by incorporating higher time intervals become more significant. In addition to lower solution times, considering larger time intervals can make it less probable for travel time uncertainties to cause missed rides (specially during transfers), since it might not be as

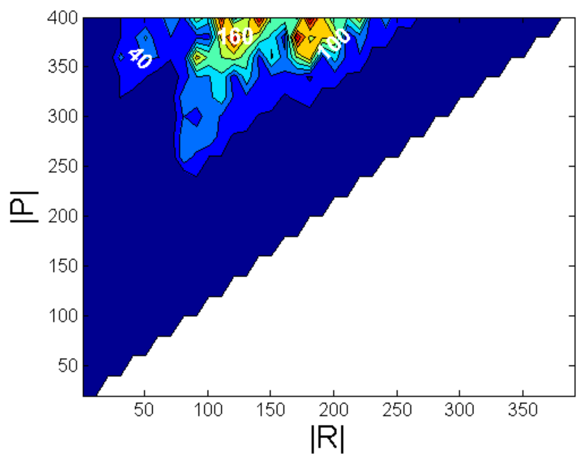


Figure 7.13: Solution times with 5-min time intervals

easy for drivers to guarantee 1 minute precision.

Another issue that needs to be discussed is the circumstances under which it is appropriate to use larger time intervals. In practice, individuals tend to report their schedules in five-minute increments at the finest, i.e., it is not typical to hear an individual planning to leave no later than 2:21 P.M., and therefore in general using larger time intervals (e.g., 5 min intervals) seems appropriate. For a ridesharing system that is integrated with a transit system with minute-level time schedule, however, using smaller time intervals might be more suitable.

7.5.4 Heuristic Solutions

In each iteration of the decomposition algorithm, a solution to the original ride-matching problem is available. In section 7.3.2, we saw that we can obtain a heuristic feasible solution in each iteration of the decomposition algorithm by solving a set packing problem. In this section, we study the impact of stopping the decomposition algorithm at certain time limits, and retrieving the best heuristic solution. The results are displayed in Figure 7.14.

Figure 7.14 suggests that the additional number of served riders obtained from letting the computations continue beyond 240 seconds is negligible. Comparing Figures 7.14 and 7.7

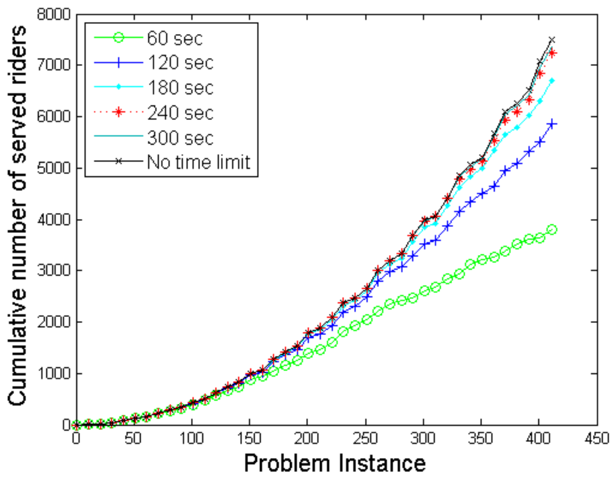


Figure 7.14: Quality of heuristic solutions

suggests that even at a computational cut-off time of 120 seconds, the number of served riders is higher than or comparable to the number of served riders in the other four ride-matching methods described in section 7.4.2. In fact, for several of the problems, even 60 seconds is sufficient. The fact that computations can be accomplished in a mere 2-minute period again validates the applicability of the optimizations schemes in dynamic settings.

Chapter 8

P2P Ride Exchange Mechanism

8.1 Introduction

Customers in many transportation systems are served on a first-come first-served (FCFS), or otherwise pre-ordered basis. For P2P ridesharing in which customer retention is especially important, considering riders on a FCFS basis is an inefficient use of the very limited available resources (drivers). The FCFS rule, however, is the natural order of serving riders in a dynamic system, where riders announce their trips not long before departure. In addition, dropping the FCFS principle leads to high solution times for the resulting matching problem, and is therefore not an appropriate implementation strategy for dynamic real-time systems.

In this chapter, we introduce what we call the "P2P ride exchange" mechanism to improve the number of matches in a FCFS-based system. In a system where P2P ride exchange is implemented, riders will still be considered for service on a FCFS basis. Upon joining the system, a rider will be offered the best available itinerary, according to the criteria discussed in section 6.3.1 (i.e. number of transfers, and traveling and waiting times). However, if no match exists, the rider will be given the chance to buy a previously-matched rider's itinerary under specific circumstances. Purchasing an itinerary from a previously-matched rider is in fact reversing the FCFS rule. This exchange of rides is accompanied with an exchange of money through the system. Since the objective of the system from implementing the exchange mechanism is to increase the total number of matched riders, only riders for whom an alternative itinerary is available will receive a proposal to sell their current itineraries.

This chapter has been a collaboration with Roger Lloret-Batlle. Extensions of this work will appear on Roger's dissertation.

There are, however, considerable regulatory obstacles to overcome for such P2P exchange or trade schemes to be used in transportation systems. The legal battles faced by ridesharing companies are well-known. Transportation supply being considered a public good, any breaking of the traditional FCFS operational paradigms also could face objections based on socio-political arguments of inequity across users. While important, such topics are considered beyond the scope of this dissertation that focuses only on showing the performance potential of the proposed scheme.

8.2 Related Work

There have been a few attempts in the literature to design mechanisms for para-transit and ridesharing systems. Furuhata et al. (2015) propose the Proportional Online Cost Sharing Mechanism for demand-responsive transport Systems. This mechanism is capable of proposing an upper bound on the fare a potential user has to pay. The mechanism relies heavily on having the passenger requests in advance of the start time of the trips. The focus of the work is on proving the online fairness, budget balancedness, individual rationality and ex-post incentive-compatible properties of the mechanism under certain conditions. The mechanism, however, unlike the P2P ride exchange mechanism proposed in this chapter, is not designed to increase operational efficiency of the transport system. Wang (2013) proposes a stable matching game between riders and drivers in a one-to-one system, where no rider/driver can be better off by unilaterally switching to other drivers/riders. Although such a system can lead to an equilibrium, to yield operationally efficient results it requires access to the participants' trip information in advance. Kleiner et al. (2011) proposes an auction-based allocation mechanism that incorporates users' valuations on the ride assignment. While this mechanism violates the FCFS rule, it is not real-time since it uses a rolling horizon in which decisions are delayed.

P2P ride exchange is the first real-time mechanism designed to compensate, to some extent, the inherent trade-off between the two factors that influence customer satisfaction, serving higher number of riders and system responsiveness, in a ridesharing system. To the best of our knowledge, the proposed P2P ride exchange mechanism is the first trading mechanism to increase ridership in a dynamic P2P ridesharing system. The designed mechanism is limited to bilateral trade, where there is a single buyer and a single seller. Such a mechanism therefore, is optimal for a one-to-one matching system, and provides a lower-bound on the rise in ridership in one-to-many and many-to-many systems.

In the rest of this chapter, we first officially introduce the mechanism, and elaborate on some of its properties. Next, we conduct extensive numerical experiments to quantify the performance of the P2P ride exchange mechanism under different parameter values.

8.3 Peer-to-Peer Ride Exchange

Dynamic ridesharing systems should have the capability of matching riders and drivers in real-time. Since participants in a dynamic ridesharing system announce their trips not long before they are ready for departure, the attempt to find a match for them should start as soon as the trip announcement is received by running the DP algorithm described in Chapter 6. If all the itineraries generated by the algorithm are infeasible due to their conflicts with itineraries of the previously assigned riders (i.e. if the itineraries use the same drivers, but through different paths), then the system evaluates the possibility of a trade. In this section, we show through an example the benefits of a P2P exchange program, discuss the conditions under which trade can happen, and devise a mechanism that ensures a fair trade.

Let P^r denote the set of itineraries for rider r. Each itinerary has a value that is determined by a pre-specified objective function (the DP objective function), based on which the itineraries within P^r are ranked. Let p_i^r denote the i^{th} itinerary of rider r, and $d(p_i^r)$ denote the set of drivers who contribute to itinerary p_i^r. Note that there is no need to know all members of set P^r in advance, but we will generate them as (and if) needed. Furthermore, let p_k denote the itinerary of the assigned driver k.

Once rider r joins the system, the system uses the DP algorithm to generate a set of trees from which members of set P_r can be retrieved. The system starts by evaluating members of set P^r in order of their ranking. If an itinerary with no conflicts with the itineraries of previously matched drivers is found, this itinerary will be assigned to rider r. If the system exhausts all members of set P_r, and is not successful in finding a non-conflicting itinerary for rider r, then it considers the possibility of a trade.

Assume that rider 1 enters the system, and has two itineraries: $P_1 = \{p_1^1, p_2^1\}$, where $d(p_1^1) = \{d_1\}$ and $d(p_2^1) = \{d_2\}$. The left hand side picture in Figure 8.1 shows the rider and his itinerary set. Assuming that the minimum cost itinerary for this rider is the first one, this itinerary will be announced to both rider 1 and driver 1.

Now, assume that rider 2 joins the system. Because rider 1's itinerary has been fixed, there are no feasible itineraries for rider 2. However, rider 2 has a chance to buy rider 1's itinerary, if rider 1 has not started his trip yet. The right hand side picture in Figure 8.1 shows this

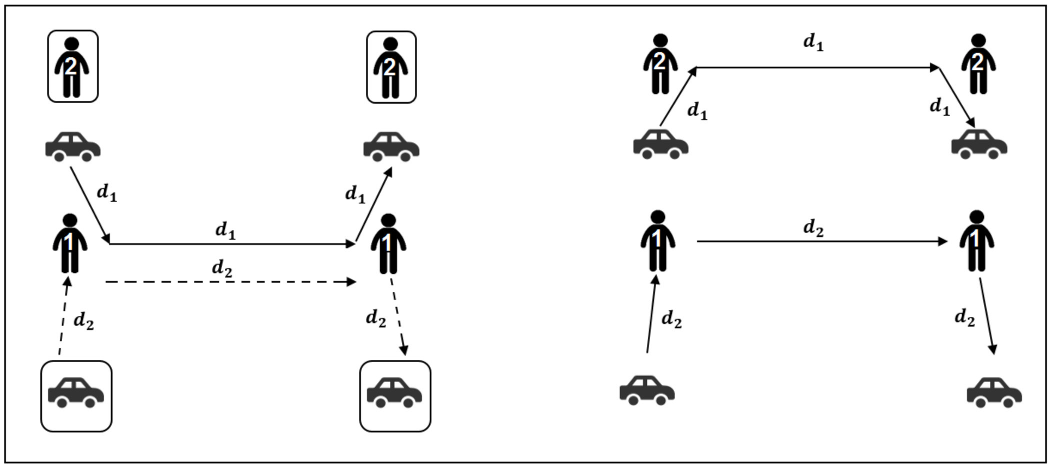


Figure 8.1: An example of a successful trade. Solid and dashed lines represent proposed and potential itineraries, respectively

scenario after the trade. In this trade, rider 2 buys rider 1's assigned itinerary, and by doing so releases driver 1, who in turn forms a feasible itinerary for rider 2. Rider 1 switches to a less convenient itinerary (with driver 2) in exchange for a monetary compensation. This trade's contribution to customer retention is double-folded. Not only are both riders served, but now both drivers are participating in the system as well.

Note that it is possible to obtain the same optimal solution by solving a many-to-many ride-matching problem that is capable of considering both riders at the same time. There are, however, two issues with such an approach: (1) An optimal matching algorithm that could consider both riders at the same time is computationally hard to solve (specially for real-world size problems), and therefore cannot yield solutions in real-time. (2) Even if the system is equipped with a many-to-many ride-matching algorithm that can yield solutions in a moderate period of time, the information on the two drivers and the two riders need to be available in advance for the many-to-many matching problem to generate the solution that can serve both riders.

The system studies the possibility of a trade if the following three conditions hold. First, the buyer does not have any feasible itineraries. Second, the seller has an alternative feasible itinerary to his current one, and third, both parties will be better off with the trade than without it.

The monetary transfer from the buyer to the seller covers the additional cost the seller has to undertake due to switching itineraries. This cost includes the additional monetary cost due to a potentially increased travel distance, and a compensation to the seller for a potentially increased travel time. A proportion of this money will be used by the system operator to cover the cost of the seller's more expensive new itinerary, and the rest will be transferred to the seller himself.

8.3.1 The Scope of the Trade

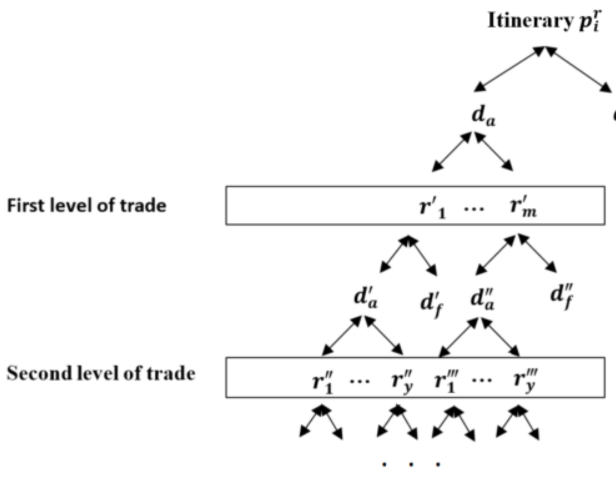
Assume a set of itineraries P^r for rider r. Drivers contributing to itinerary i are stored in set $d(p_i^r)$. Let us divide members of set $d(p_i^r)$ into two mutually exclusive sets, $d_a(p_i^r)$ and $d_f(p_i^r)$. Drivers in set d_a have been previously assigned to other riders, but their corresponding riders' trips have not started yet. Drivers in set d_f are free, and have not been assigned to any riders. The necessary condition for rider r to have a feasible itinerary is for at least one of the driver sets $d_a(p_i^r)$ and $d_f(p_i^r)$ to be non-empty. As the sufficient condition for p_i^r to be a feasible itinerary for rider r, one of the following conditions should hold: $(1) d_a(p_i^r) = \emptyset$, i.e. none of the drivers that contribute to the itinerary are assigned to other riders, and (2) $\forall k \in d_a(p_i^r), p_i^r(k) \in p_k$, i.e. drivers in set $d_a(p_i^r)$ can still follow their previously assigned itineraries. If none of these two conditions hold, then the system tries to find a good candidate for a trade amongst the assigned riders.

To find the candidates for a trade, the system has to first identify the itinerary that rider r is interested in. It starts from the best itinerary, i.e. p_1^r, and moves to the next itinerary if the trade on the current itinerary is not possible.

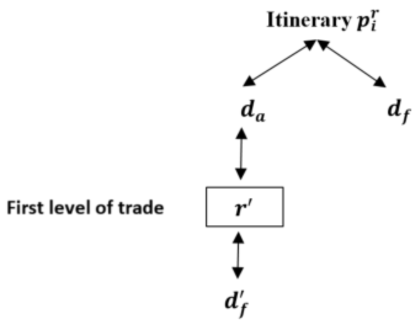
In order for the system to offer itinerary p_i^r to rider r, it has to free all the drivers in set $d_a(p_i^r)$ from their previous assignments. Therefore, the system has to find all the riders who are using these drivers, and try to find alternative itineraries for them as well. These riders form the sellers in the first level of trade (Figure 8.2a). In order for the system to propose an exchange to a rider r' in the first level of trade, it should find an alternative itinerary for this rider first. This task can be accomplished by identifying the set of assigned drivers for the rider (d'_a), finding the rest of the riders whose itineraries are affected by these drivers, and finally finding alternative itineraries for them as well. This procedure continues until the system reaches a level of trade where all riders have itineraries with free drivers (or non-conflicting assigned drivers).

The system will then start proposing trades to riders, starting from those in the last level of trade. In order for a trade to be approved at any level, all the riders at that level should approve the trades proposed to them. For the n^{th} level of trade to take place, the trade at level $n+1$ should have been approved. Once all riders in a given level of trade approve the proposed trades, the system can move to a higher level of trade (moving upward in Figure 8.2a). Therefore, it is clear that the higher the levels of trade, the less likely it is for rider r to obtain itinerary p_i^r.

Another complication is that if even one rider does not agree to the trade at a certain level, the trade cannot happen. In this case all the riders in the same and lower levels who have agreed to the trades proposed to them have to go back to their previous itineraries. Therefore, in order to simplify this procedure and make it easy to implement in practice, this chapter only considers trades in settings where the level of trade is limited to 1, and the number of riders in the first level of trade is limited to 1 as well, i.e. the set of assigned drivers affect only the itinerary of a single previously assigned rider (Figure 8.2b). These simplifications limit the trade to be between on two individuals: the buyer, r, and the seller, r'.



(a) General levels of trade

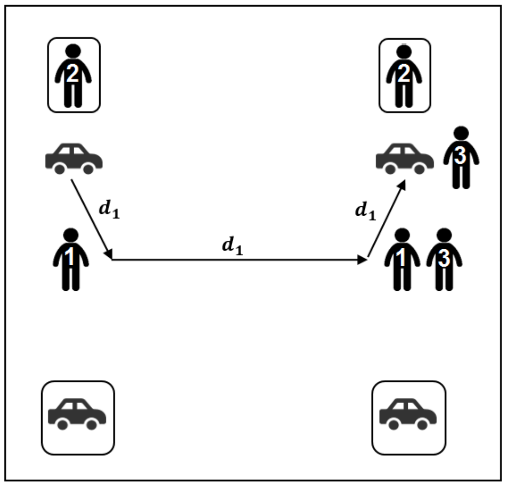


(b) Level of trade covered in this study

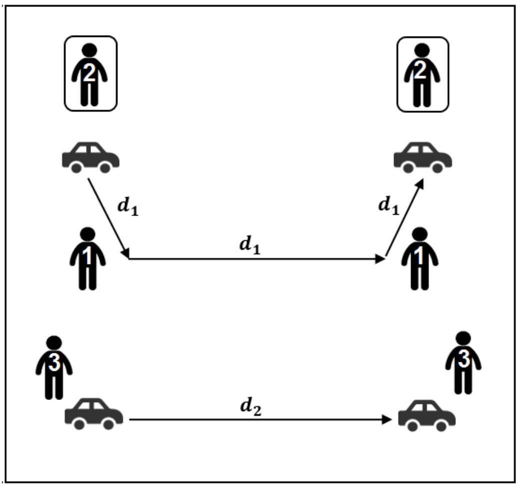
Figure 8.2: Levels of trade

Figure 8.3 displays two examples involving multilateral trade. In the first example (Figure 8.3a), in order for the system to serve rider 2 by freeing driver 1, it must find alternative itineraries for both riders currently served by driver 1 (riders 1 and 3). For this to happen, rider 2 should negotiate with both riders 1 and 3, which is beyond the scope of the bilateral trade covered in this chapter. Notice that this example is still limited to the first level of trade in Figure 8.2a. Furthermore, even if a multilateral trade mechanism was available, in order for the trade to happen, both riders 1 and 3 should have agreed to the trade.

Figure 8.3b demonstrates an example of a simple scenario involving riders beyond the first level of trade. In this example, two alternative itineraries are available for rider 1, just as in Figure 8.1. When rider 2 joins the system, he is interested in purchasing the itinerary assigned to rider 1. In this example, however, driver 2 has been matched with rider 3, who belongs to the second level of trade. Therefore, for the trade to happen, the system should find an alternative itinerary for rider 3 first.



(a) More complex trade at the first level



(b) Trade at the second level

Figure 8.3: Examples of multilateral trade

8.3.2 P2P Ride Exchange Mechanism

Besides ensuring that the trade makes both parties better off, the designer (operator) should also ensure that the trading parties cannot manipulate the outcome of the trade. Since both the buyer and the seller hold private information not known to the operator (e.g. their value of time (VOT)), this could lead to an inefficient outcome. This issue is addressed by modeling the trade from a mechanism design perspective. Informally, a mechanism is a method that defines rules for a game with incomplete information (Bayesian game) to influence agents' behavior and reach a particular goal, which in this case is efficiency maximization. Excellent introductions to mechanism design can be found in Mas-Colell et al. (1995) and Nisan and Ronen (1999). The basic definitions are provided next, but a complete understanding of mechanism design may require reading the above introductory references.

Let $I = 1, \ldots, n$ be the set of agents. Each agent has a type (value of time), $\theta_i \in \Theta_i$, which is private. $\Theta = \times_{(i \in I)} \Theta_i$ is the type profile set. Agent i has the (quasilinear) utility function $u_i(\theta_i, \theta_{-i}; \theta_i) = v_i\big(k(\theta_i, \theta_{-i}); \theta_i\big) - p_i(\theta_i, \theta_{-i})$. Where $v_i(\cdot)$ is his valuation and $p_i(\cdot)$ is the price charged to him. The types before the semicolon are the types announced to the designer, while the type at its right is the agent i's actual type. A (direct revelation) mechanism is composed of two interrelated functions. Firstly, an allocation function $k : \Theta \rightarrow K$ that maps the type space to an outcome set K. That is, for every announced type profile, an allocation $k(\theta) \in K$ is given. In the case of a bilateral trade, K is composed of the two allocations (trading states): either there is trade or there is not. The allocation rule that maximizes the sum of agents' valuations is the efficient allocation rule, $k^*(\theta) \in K$. Secondly, there is a payment function $p : \Theta \rightarrow R^N$. This function assigns a transfer amount to every agent i in accordance with its announced type θ_i.

Mechanism design defines concepts that address how the strategic interests of agents are satisfied. The main one is truthfulness, or incentive compatibility, which states that truthful bidding forms an equilibrium. In other words, any participating agent is always better off by truthfully eliciting its type rather than lying, subject to others telling the truth. A mechanism (k, p) is (Dominant-Strategy) Incentive Compatible (DSIC) if all agents are better off (or at least not worse off) by being truthful, regardless of other agents' behavior.

Besides truthfulness, a designer is interested in the users' willingness to participate in the mechanism, called individual rationality. A mechanism (k, p) is Ex-Post Individual Rational (EPIR) if no agent loses due to participating in the system.

where $\bar{u}_i(.)$ is agent i's utility from not participating in the mechanism. If EPIR is satisfied, an agent would be willing to truthfully participate in the mechanism rather than stay out. Finally, the designer may be interested in the mechanism being self-sufficient from a budget point of view. In this way, the mechanism should not require an external subsidy to achieve the desired outcome. This condition is known as balanced budget. A mechanism (k, p) is strictly budget balanced (SBB) if it is financially independent, and does not require subsidies to operate.

We follow a "pessimistic" approach (Yamashita, 2015) in which the designer calculates the expected surplus that is guaranteed among all the admissible strategy profiles of the agents, with the actual strategies played being unknown to the designer. A strategy is said to be admissible if it is not weakly dominated. It is assumed that the operator has a prior on the private information from agents, but the agents themselves do not have a prior of the other agents' type, as in the case of weaker truthfulness concepts (Myerson and Satterthwaite, 1983). This framework is very convenient for our purposes, since the mechanism has to be designed far in advance, with no previous experience or learning on trading outcomes from either the users' or the designer's part, while, at the same time, it guarantees an increase in the number of served riders to achieve user permanence in the system.

The trade is modeled as a bilateral trade with private information (Hagerty and Rogerson, 1987; Myerson and Satterthwaite, 1983). Let $I = 1, 2$ be the set of agents, $i = 1$ being the seller and $i = 2$ being the buyer. Each agent i has type $v_i = [\underline{v_i}, \overline{v_i}]$. These types are drawn from an empirical VOT distribution estimated from a survey on households conducted in Stockholm, Sweden in 2005 (Abou-Zeid et al., 2010). In that research, the Stated Preferences (SP) choice scenarios are composed of car alternatives that differ on attributes such as travel times and travel costs. Since only the main statistics are available in the publication, the distribution is recalibrated as a lognormal distribution given these statistics. Its parameters are location $\mu = 2.16$ and scale $\sigma = 0.40$.

The mechanism lies in the space $(q, p) \in [0, 1] \times R$, where q is the probability of trade and p is the payment from the buyer to the seller. For clarity in the exposition, we use the following change in notation $c_1 \overset{def}{=} -v_1$. c_1 is the opportunity cost of the seller. By definition, the bilateral trading setting satisfies the strict budget balance property, thus the seller has utility $u_1 = p - c_1 q$ and the buyer $u_2 = v_2 q - p$. Both agents are proposed the price p and if both agree, the trade takes place. This occurs when $v_2 > p > c_1$. The surplus of such as trade is $w\big((q, p); \theta\big) = (v_2 - c_1)q$.

Instead of valuing an object by a scalar as in the original bilateral trading environment, riders value their allocation (assigned ride) by its generalized cost, which is an affine transformation of their private type. For a rider i, this cost is the product of the travel time t_{ri} and the sum of the value of time θ_i, plus the fare per unit of time c_{ri}. These valuations are normalized with regard to the initial situation (no trade) to fit the bilateral trading original setting: c_1 and v_2 are in fact the valuation difference between states "trade" and "no trade". When there is a trade, $c_1(\theta_1) = \theta_1(t'_{r1} - t_{r1}) + c'_{r1}t'_{r1} - c_{r1}t_{r1}$ and $v_2(\theta_2) = \theta_2(t_{out} - t_{r2}) + c_{out}t_{out} - c_{r2}t_{r2}$. Here, t_{r1}, t'_{r1}, t_{r2}, and t_{out} refer to the travel time of the seller's current and new itineraries, travel time of the buyer's itinerary, and the travel time of the buyer's outside alternative, respectively. c_{r1}, c'_{r1}, c_{r2} and c_{out} are the costs per unit time of seller initial ride, seller alternative proposed ride, buyer proposed ride and the outside option cost to the buyer.

These time and cost variables have bounds and relative magnitudes. $t_{out} \leq t_{r2}$ since we consider the outside option to use the shortest path between buyer's origin and destination. We assume that cost of the rideshare option to the buyer is less than that of the outside option, i.e. $c_{out}t_{out} - c_{r2}t_{r2} \geq 0$; Otherwise the buyer would not have selected to use the rideshare option. This assumption leads $v_2 = [\underline{v_2}, \overline{v_2}]$ to be positive in our analysis. Note that as θ_2 increases, v_2 decreases, so $\underline{v_2} = max\big(0, v_2(\underline{\theta_2})\big)$ and $\overline{v_2} = v_2(\overline{\theta_2})$. Since the seller is offered a longer ride than the one he holds, $t'_{r1} \geq t_{r1}$ and $c'_{r1}t'_{r1} - c_{r1}t_{r1} \geq 0$, with a high probability. If due to higher number of riders involved in the new itinerary c'_{r1} becomes smaller than c_{r1}, the system will set $c'_{r1}t'_{r1} - c_{r1}t_{r1} = 0$. In this way, $c_1 \in [\underline{c_1}, \overline{c_1}] \geq 0$ and it is increasing with θ_1. Its bounds are $\underline{c_1} = c_1(\underline{\theta_1})$ and $\overline{c_1} = c_1(\overline{\theta_1})$. Without loss of generality, we assume $\underline{\theta_1} = \underline{\theta_2} = \underline{\theta}$ and $\overline{\theta_1} = \overline{\theta_2} = \overline{\theta}$. Trade is only possible when $\underline{v_2} \geq \overline{c_2}$.

The problem that now the designer faces is to post a price to agents such that the probability of trade and consequent surplus is maximized.

Proposition 1. *The posted-price mechanism with price p for the bilateral trade setting is a* *revelation mechanism (q, p) such that:*

$$(q,p) = \begin{cases} (1,p) & v_2 > p > c_1 \\ (0,0) & otherwise \end{cases}$$

This mechanism is DSIC, EPIR, SBB and guarantees the following upper bound expected surplus $W(p)$:

$$W(p) = \iint_{(c_1,v_2)=\left(c_1(\underline{\theta_1}),\tau\right)}^{(c_1,v_2)=\left(\tau,v_2(\underline{\theta_2})\right)} (v_2 - c_1)\phi(c_1, v_2) dv_2 dc_1$$

Proof. Let the strategy of the seller s_{seller} be: sell if $p > c_1$ and do not sell if $p \leq c_1$. This is the only DSIC strategy for the seller (we assume that the seller prefers to keep the object when the gain is zero). Suppose that the seller follows a different strategy s'_{seller}: sell only when $p - c_1 > \epsilon, \epsilon > 0$. Then the seller would lose the opportunity to make profit $p - c_1$ when $c_1 + \epsilon > p > c_1$. Equivalently, when ϵ is negative, the seller would incur a loss of $c_1 - p$ when $c_1 + \epsilon < p < c_1$.

Moreover, s_{seller} is the only strategy that is EPIR. Seller's payoff of not participating in the mechanism is zero. Following the same reasoning done above on incentive compatibility, s_{seller} is the only strategy that provides a non-negative profit for every buyer and seller type. The same reasoning applies for the buyer, with strategy s_{buyer}: buy if $v_2 > p$ and do not buy if $v_2 \leq p$.

The mechanism is SBB by construction. When there is no trade the transfer is zero. When there is trade, the positive price payed by the buyer goes to the seller. There is no waste in the numeraire in any case.

The expected welfare $W(p)$ is the integral of welfare function weighted by the joint type probability distribution function. The bounds determine the entire valuation range over which trade happens and therefore when there is positive surplus from trade. When trade does not happen, the surplus is zero. Each expected surplus integral is solved by numerical simulation in price intervals of 5 cents. The optimal price p^* is found by linear search. Since the VOT distributions of the agents are assumed to be independent, the VOT joint distribution is the product of the two marginal distributions. Empirical value of time distributions do not satisfy the sufficient condition stated in (Yamashita, 2015) since they are neither monotonically increasing/decreasing in buyer and seller type respectively over all the range nor the rate of change is small enough. Therefore, $W(p^*)$ stays a priori as the upper bound of the highest expected surplus we can guarantee. $\square$

8.3.3 Pricing

There are many factors that should be taken into consideration in determining the fare for ridesharing services. Setting the right price is essential to the success of a ridesharing system, and deserves the designing of a separate mechanism which ensures that no incentive exists for drivers and riders to falsely report their preferences in order to affect the amount of transaction.

The fare a rider is charged in our system is made of two components. The first component is a variable, distance/time dependent fee. Assume that rider r's itinerary involves traveling on link set L^*, and that at the time of matching the rider, on each link $\ell \in L^*$, n_ℓ number of individuals (including the driver) share the same vehicle with the rider. The cost of travel on each link is equally shared by the individuals who travel on the link. Therefore, the variable fee of rider r will be $\sum_{\ell \in L^*} \frac{d_\ell}{n_\ell}$. In this equation, d_ℓ is the general cost of traveling on link ℓ, and $\sum_{\ell \in L^*} \frac{d_\ell}{n_\ell}$ is the total share of the rider from the cost. Note that the general cost of a link can be time-based, distance-based or a combination of both.

In addition to a variable component, the fare also has a fixed component. Since drivers may have to divert from their shortest/preferred paths in order to accommodate riders, they need to be compensated for the extra travel. We calculate the base fare based on the average extra travel time drivers have to spend in the network, assuming an average speed of 40 mph, and a payment of 60 cents per mile. These fares could vary for different times of the day, and days of the week, based on the composition of the ridesharing system, i.e. number of participants, the driver to rider ratio, and the degree of flexibility of riders and drivers. Although a pricing scheme that can distribute fares among drivers based on their contribution to the system may be fairer, in the interest of simplicity we use the more preliminary pricing scheme introduced in this section.

8.4 Numerical Study

In order to study the impact of the P2P ride exchange mechanism on the performance of a ridesharing system, we generate and solve multiple random instances of the ridesharing problem. All results reported here are averaged over 30 runs for each problem instance.

In each problem instance, we generate a number of participants with varying ratio of riders. The origin and destination of participants are selected based on a uniform random distribution from a pre-specified set of stations in a grid network. The earliest departure time of each participant is selected uniformly by a random distribution within a certain departure period. The maximum ride time of each participant is determined as a factor of their shortest path travel time (called "travel time budget factor"). The latest arrival time of a participants at their destination station is then computed as the sum of the participant's earliest departure time and maximum ride time. All these parameters impact the level of spatiotemporal proximity between trips.

For each participant, a VOT is drawn from the lognormal distribution described earlier. Each individual is assumed to have a separate transportation alternative outside of the system, with a travel time equal to the shortest path travel time between the individual's origin and destination. The unit distance-based cost of the outside alternative is assumed to be equal to that of the ridesharing system.

We solve each problem instance using three different ridesharing implementation strategies. The first strategy referred to as "one-to-one", matches a single rider with a single driver. The second strategy referred to as "one-to-many" allows a driver to carry multiple riders. Riders, however, accomplish their trips in one vehicle. The last implementation strategy referred to as "many-to-many" allows each driver to carry multiple riders, and each rider to transfer between multiple drivers. Note that the P2P exchange mechanism is optimal only for the "one-to-one" matching method. The number of additional riders served by the exchange in "one-to-many" and "many-to-many" systems is only a lower bound and may increase using a more sophisticated mechanism that can include higher levels of trade.

In this section, we perform sensitivity analysis over the system parameters, namely number of participants, ratio of riders, travel time budget factor, and number of stations. Through these analysis we study the impact of different parameter values on the percentage increase in the number of matched riders (to which we refer as the *exchange rate*). Finally, we generate different ridesharing scenarios with different levels of spatiotemporal proximity between trips, and use statistical tests to confirm whether the observed difference in the exchange rates in these scenarios is statistically significant.

8.4.1 Base Fares

In order for a rider to decide whether to participate in a trade or not, he/she should have information on the cost of the proposed itinerary. As discussed in section 8.3.3, the cost of an itinerary entails a variable, route-dependent cost, and a fixed cost. In this section, we demonstrate for certain parameter values how this fixed cost is calculated, and how it may differ from hour to hour or day to day.

Figure 8.4 displays the base fares charged to riders and payed to drivers in a ridesharing system at different degrees of spatiotemporal proximity between trips. As mentioned in the previous section, this study uses the same base fares for all drivers and a different but equal base fare for all riders in a given time period, for example, during weekday morning peak hours. These fares may vary from location to location, and depend on the number of participants and system composition (number of participants, and ratio of riders to total number of participants). In this section, we show sample base fares for a system with 200 participants with departure period of 60 minutes and travel time budget factor of 1.5, under different ratio of riders and number of stations.

As Figure 8.4 suggests, for a one-to-one system, the fare paid by riders is similar to the fare received by drivers, since the number of served riders and matched drivers in a one-to-one system are equal. In a one-to-many system, in which each driver carries multiple riders, the fixed fare paid by riders decreases as the ratio of riders to participants increases, since now multiple riders are served by a driver and hence they each pay a portion of the fixed fare the driver receives. Interestingly enough, in a many-to-many system, when the ratio of riders in the problem is small, each rider pays more than what each driver receives, suggesting that riders are receiving multi-hop itineraries (i.e. transferring between drivers), specially when the number of stations is high and therefore the spatial proximity between trips is too low. After a certain point, however, the fare received by each driver surpasses the fare paid by each rider, suggesting a high level of sharing.

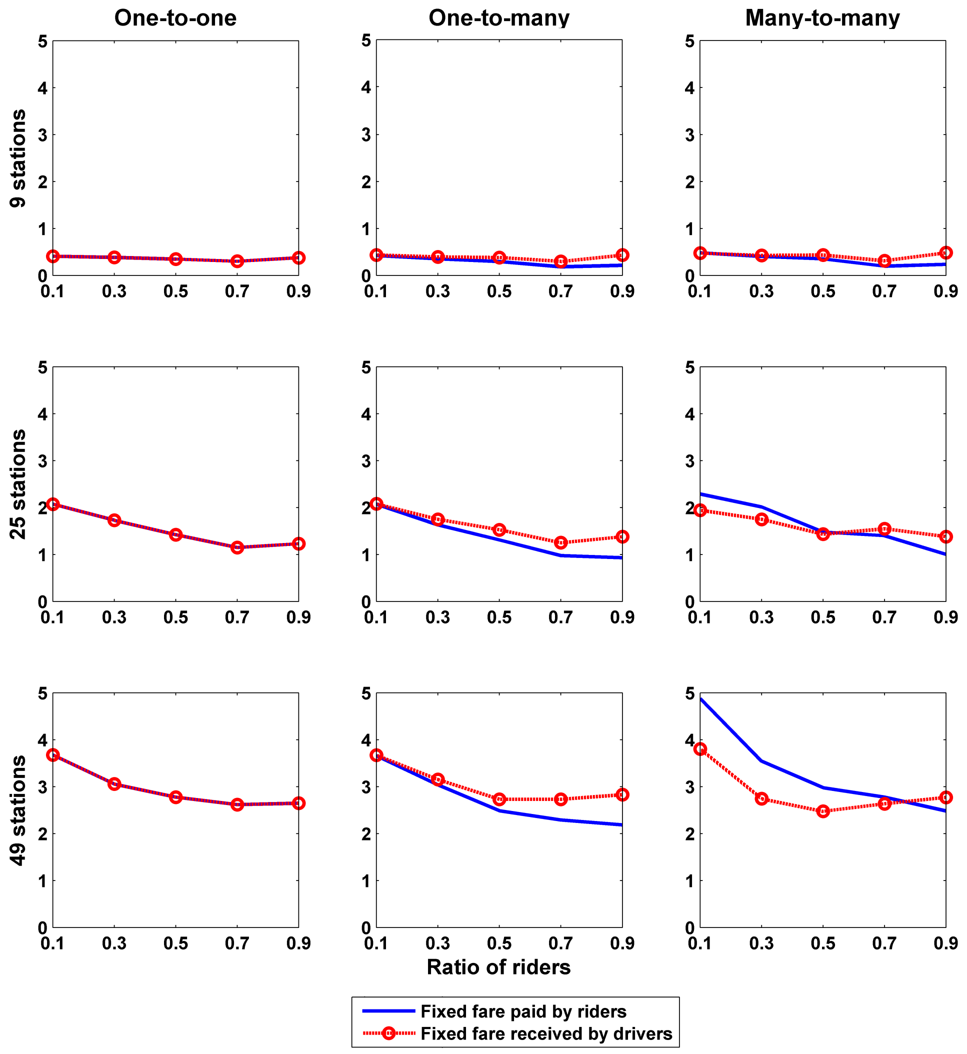


Figure 8.4: Base fares for ridesharing systems with different levels of spatiotemporal proximity between trips

8.4.2 Number of Participants

In this section, we study the performance of the P2P ride exchange mechanism under different participation rates, and different numbers of stations. The problem instances have been generated in grid networks of different sizes (9, 25 and 49 stations), departure period of one hour, and 200, 300, and 500 number of participants. For a given number of participants, number of riders and drivers are assumed to be equal, since a rider to driver ratio of close to 1 is where a ridesharing system yields the highest matching rate (see Chapter 7). Figure 8.5 shows the initial matching rate of riders, and the exchange rate under different implementations of the system.

The results suggest that under all implementation strategies, both the initial matching rate and the exchange rate are positively correlated with the participation rate, i.e. the higher the number of participants, the higher the performance of the system and the exchange rate.

Another general observation is that with all matching methods, lower number of stations results in higher initial matching rate, and higher exchange rate. This result is intuitive, since participants have to choose their origins and destinations from the set of stations. Therefore, a lower number of stations results in more participants sharing the same origin and/or destination stations, and therefore higher spatial proximity between trips. Higher spatial proximity also suggests a higher probability of finding a driver that can serve the seller in the ride exchange, and therefore higher success rate in the exchange.

Despite this general trend, the impact of spatial proximity depends on the number of participants as well. For a one-to-one system with 300 participants, the exchange rate increases as we increase the spatial proximity by moving from from 49 to 25, and finally 9 stations. With 500 participants, however, the exchange rate does not go down when we decrease the spatial proximity among trips by going from 9 to 49 stations. Even when we have 49 stations, the exchange rate does not start plummeting, since a higher number of participants by itself increases the proximity between trips to some degree.

The one-to-one matching method is the only method for which the result of the exchange mechanism is optimal; Therefore, the matching rate for such a system is the highest. Another reason why the idea of P2P exchange may serve one-to-one ridesharing systems well is that since in such systems each driver serves a single rider (if matched), purchasing a driver's itinerary would require negotiations with a single seller, and therefore is practically more likely to happen.

In one-to-many and many-to-many systems trades typically expand beyond the first level, and therefore are not accomplished. This is one reason behind the lower exchange rates for these systems, compared to the one-to-one system. Another reason for the lower exchange rate in one-to-many and many-to-many systems is that the exchange rate measures the percentage increase in the matching rate. Even if the same additional number of rider are served by the exchange mechanism under all implementation strategies, since the initial matching rates in the many-to-many and one-to-many systems are higher, they will have lower exchange rates.

Figure 8.6 shows the driver matching rates and the impact of the exchange mechanism on the increase in the percentage of matched drivers. In general, this figure follows similar trends to Figure 8.5. In a many-to-many system, the driver exchange rates are higher than the rider exchange rates, which implies a high level of sharing before the exchange.

8.4.3 System Composition

The problem instances in this section include 500 number of participants generated uniformly during a departure period of one hour, in grid networks with 25 stations. Number of riders are changed from 50 to 450 (and drivers from 450 to 50) in 100 increments, in order to study

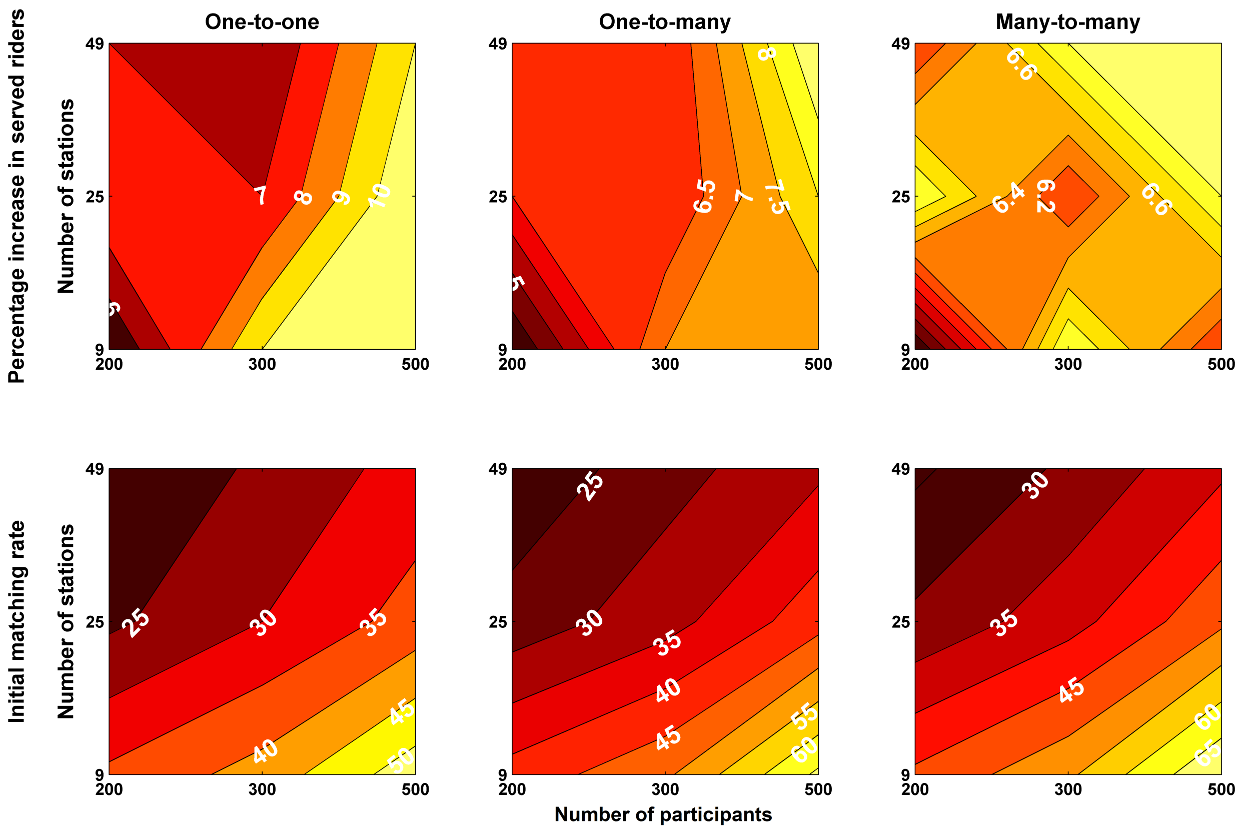


Figure 8.5: Initial rider matching and exchange rates under different number of stations and participants

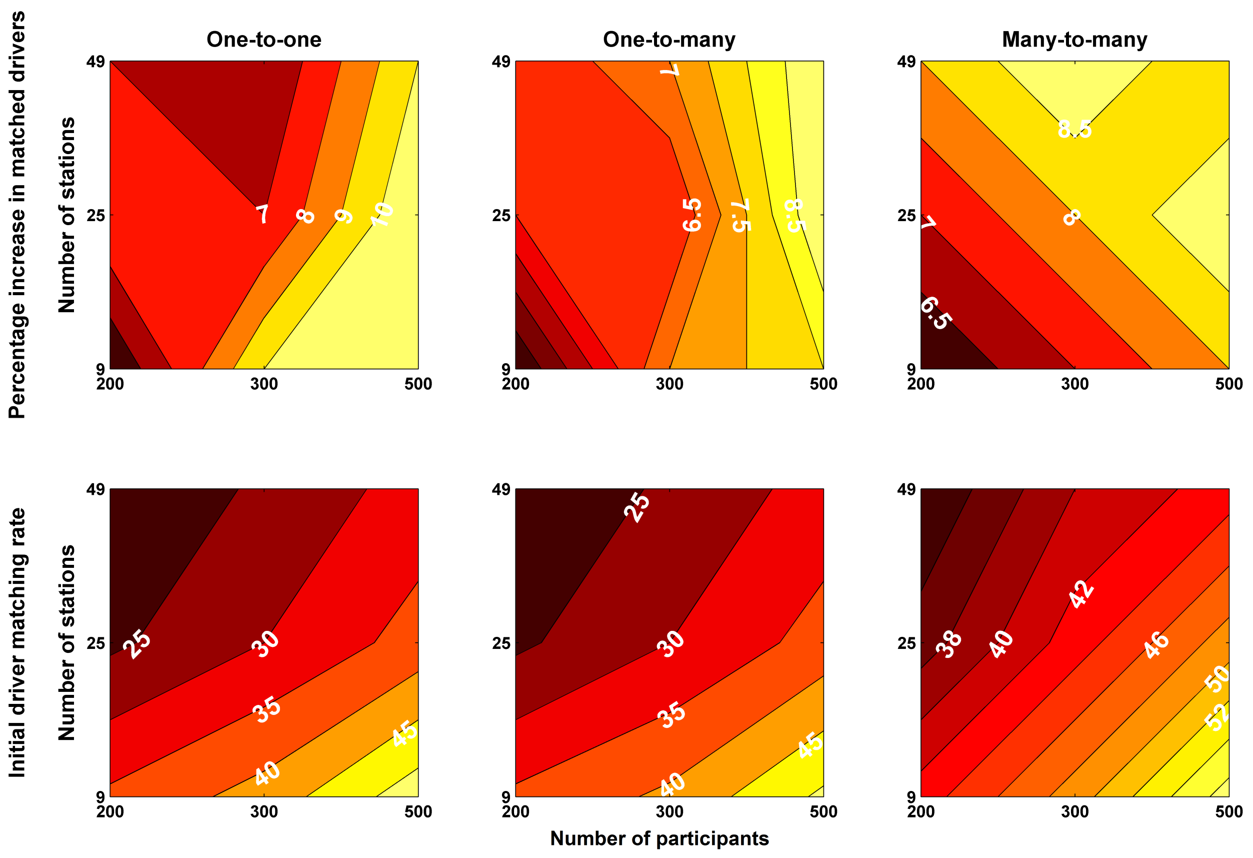


Figure 8.6: Initial driver matching and exchange rates under different number of stations and participants

the impact of system composition on the performance of the exchange mechanism.

Figure 8.7 summarizes the results. For all ridesharing implementations, as the ratio of riders to participants increases (and ratio of drivers to riders decreases), the rider matching rate experiences a declining trend. At rider ratio of 0.1 (50 riders and 450 drivers), most of the 50 riders can be matched, due to the abundance of supply (i.e. drivers). As we move in the direction of the horizontal axes, the increase in demand is met with a decrease in supply which leads to a decreasing trend in the rider matching rate. At rider ratio of 0.9 (450 riders and 50 drivers) the matching rate of riders is the lowest, due to the lack of sufficient supply.

The trend in the driver matching rate is the opposite of the trend in the rider matching rate. At lower rider ratios where there are a few riders and many more drivers, the driver matching rate is low, since a low percentage of drivers is enough to satisfy the demand. As the rider ratio increases, the demand becomes higher than supply, and so the driver utilization rate increases.

Another general trend among matching rates is the bell-shaped form of the rider and driver exchange rates. In the beginning, when the rider-to-driver ratio is low, a high percentage of demand is satisfied, and there is not much need for exchange, hence the low exchange rate. At high ratios of riders to system participants, the rider matching rate is small, but the driver matching rate is high, and therefore there are not many free drivers left to form alternative itineraries for the sellers. In the case of a many-to-many system, an exchange mechanism that can extend to higher levels of trade could help in raising the declining exchange rate. In the middle range, at rider ratio of 0.5 for the one-to-one and one-to-many systems, and 0.3 for the many-to-many system, the exchange rate becomes the highest. In this range, the rider matching rate is not too high to eliminate the need for exchanges, and the driver matching rate is not too high to decrease the chance of finding alternative itineraries for the sellers.

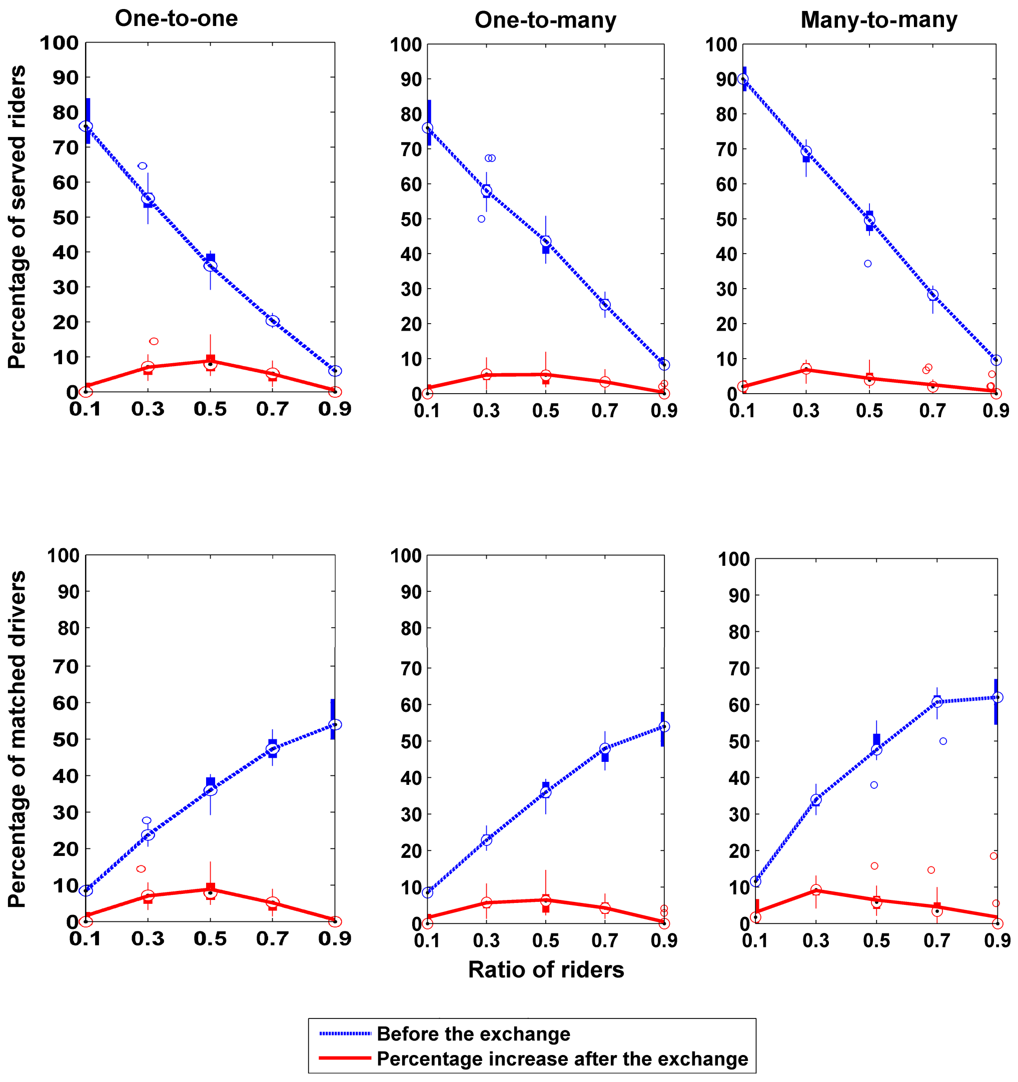


Figure 8.7: Matching and exchange rates under different ratio of riders

8.4.4 Departure Period

Similar to the previous section, the problem instances in this section involve 500 participants, with equal number of drivers and riders. The participants are assumed to have a travel time budget factor of 1.5, and their origins and destinations are randomly selected from 25 pre-specified stations. We increase the departure periods of individuals from 15 minutes to 60 minutes, in order to study the impact of higher temporal proximity between trips on the performance of the exchange mechanism.

Figure 8.8 shows that under all ridesharing implementations, both the initial rider and driver matching and exchange rates increase with the temporal proximity between trips (i.e. as the departure period becomes smaller). This is not surprising, since higher temporal proximity between trips leads to higher probability of finding a match in the first place, and higher probability of finding alternative itineraries for the sellers in case an exchange is required.

8.4.5 Travel Time Budget Factor

Experiments in this section include 200 participants, with equal number of riders and drivers, and a departure period of 60 minutes. Figure 8.9 shows the initial rider matching rate, and the percentage increase in the number of served riders under different travel time budget factors for participants. This figure suggests that in general the matching rate as well as the exchange rate increase with the travel time budget factor, regardless of the change in the proximity of trips. Figure 8.10 suggests the same results for drivers.

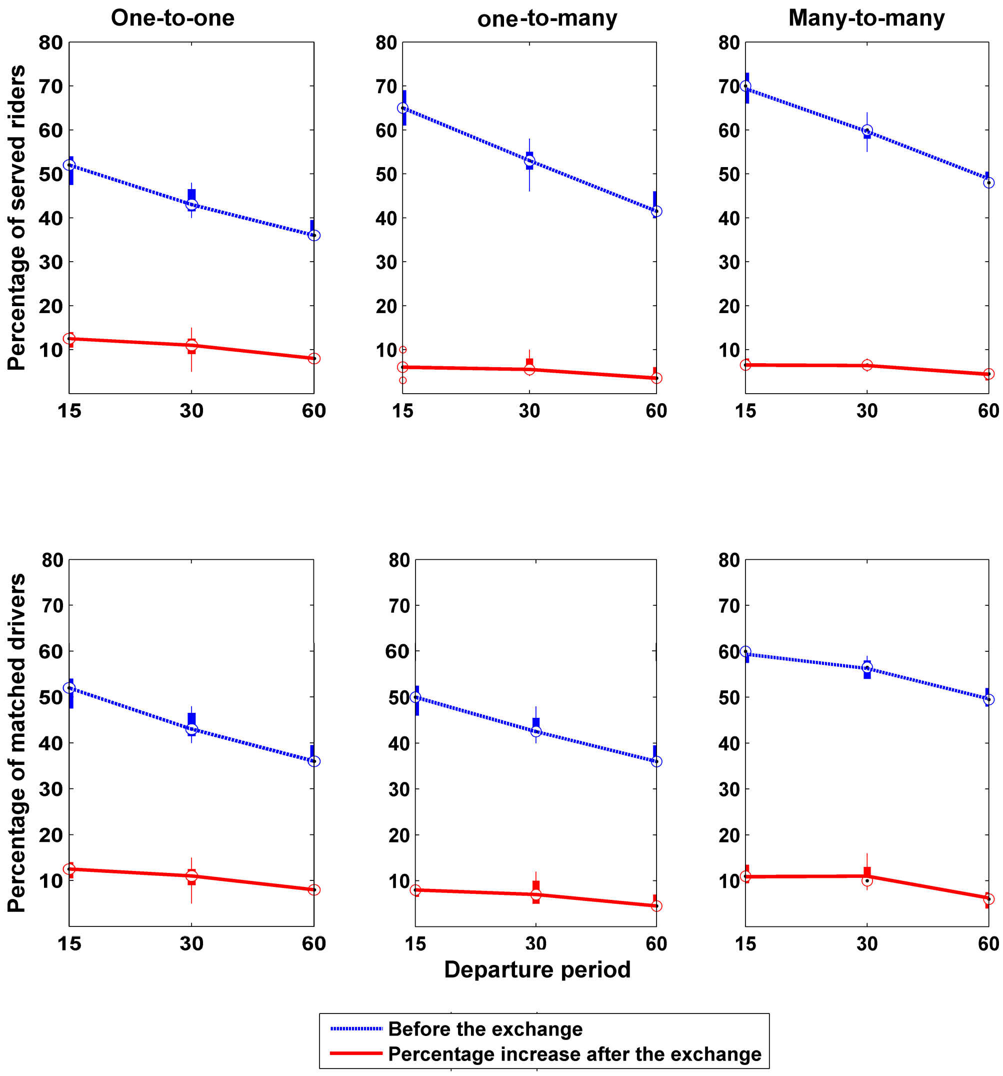


Figure 8.8: Percentage of riders and drivers matched before and by P2P ride exchange under different departure periods

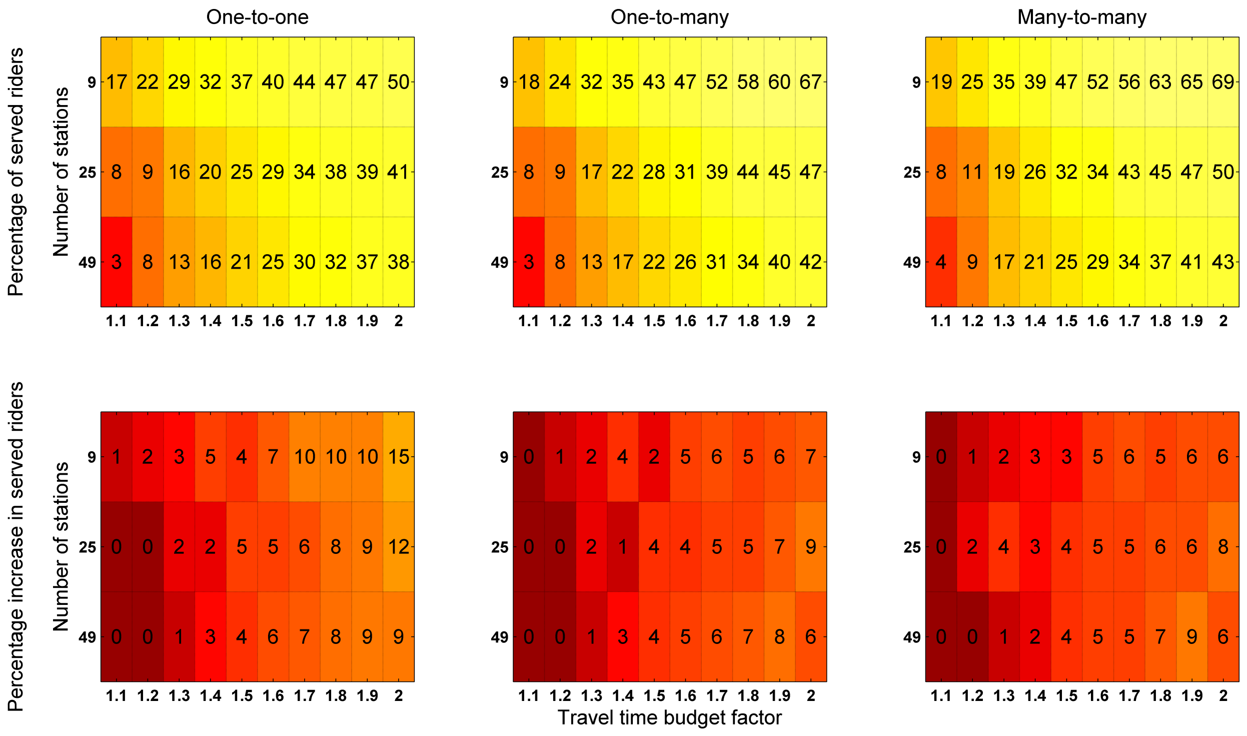


Figure 8.9: Rider matching and exchange rates as a function of travel time budget factor of participants

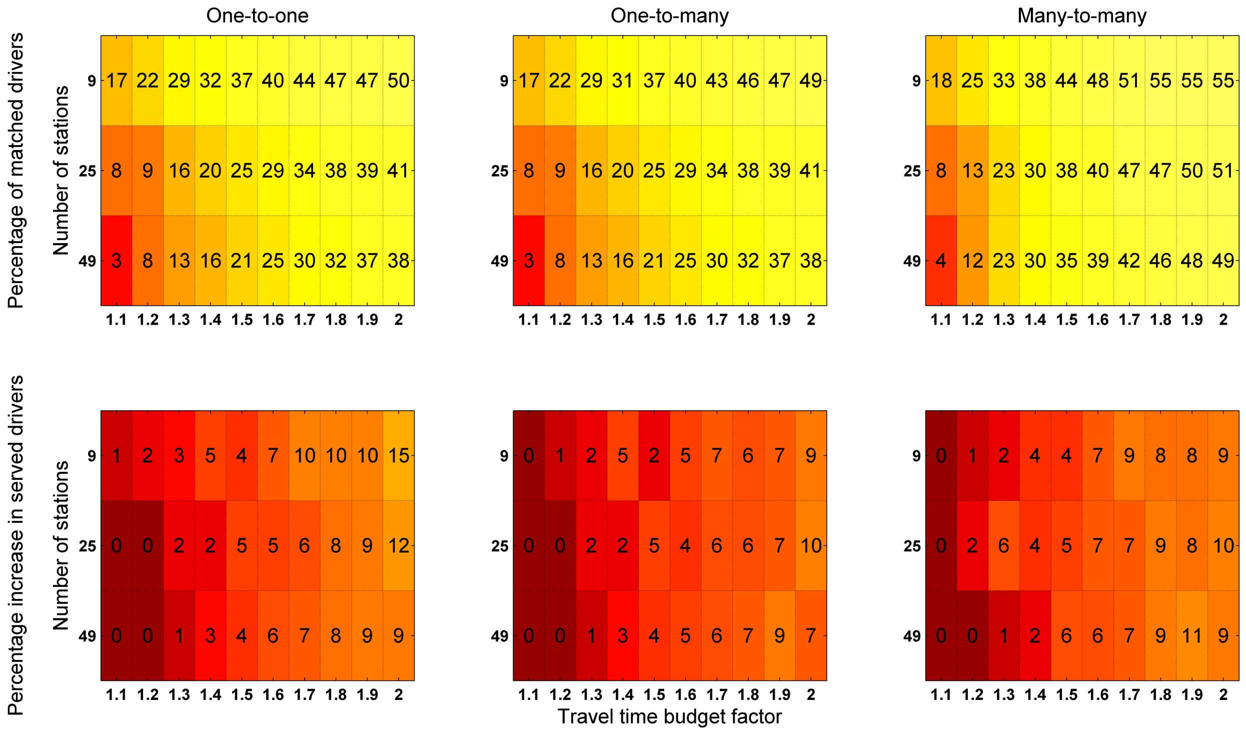


Figure 8.10: Driver matching an exchange rates as a function of travel time budget factor of participants

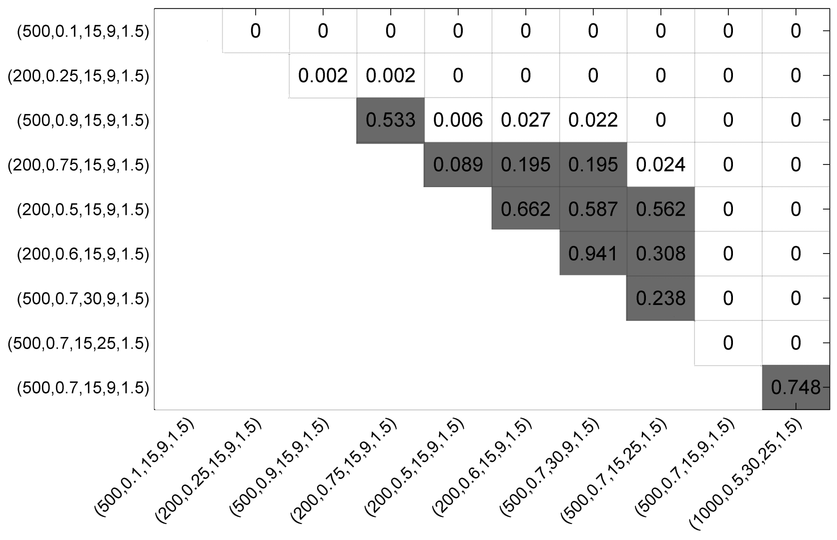
Table 8.1: Ridesharing instances. The scenario properties include (no. of participants, ratio of riders, departure period, no. of stations, travel time budget factor)

Scenario properties	Per. of served riders (mean,st.dev.)	Per. increase in served riders (mean,st.dev.)	Per. of retained riders	Average social surplus
(500,0.1,15,9,1.5)	(99,1.5)	(0.2,0.4)	0.6	11.33
(200,0.25,15,9,1.5)	(9,0.6)	(4.5,3.2)	11.5	10.92
(500,0.9,15,9,1.5)	(10,0.4)	(7.2,3.6)	18	8.82
(200,0.75,15,9,1.5)	(8,0.7)	(8,4.9)	20	8.66
(200,0.5,15,9,1.5)	(13,1.2)	(10,3.7)	26	9.69
(200,0.6,15,9,1.5)	(12,0.9)	(10,4)	25	9.47
(500,0.7,30,9,1.5)	(25,1.3)	(10,3)	28	9.01
(500,0.7,15,25,1.5)	(21,1.3)	(11,3.2)	28	14.34
(500,0.7,15,9,1.5)	(33,1)	(15,3)	38	9.97
(1000,0.5,30,25,1.5)	(60,1.6)	(15,1.2)	32	15.57

8.4.6 Statistical Analysis

Table 8.1 lists 10 ridesharing scenarios sorted in an increasing order of the exchange rate. Scenarios have different levels of spatiotemporal proximity between trips, and are all generated for a one-to-one matching method, since it is the only matching method for which the exchange mechanism is optimal.

As demonstrated in the previous sections, when the spatiotemporal proximity of trips is too low or too high, the exchange rate is small. When the spatiotemporal proximity between trips is too low, the initial matching rate is too low, and so is the exchange rate due to lack of supply. When the spatiotemporal proximity is too high, a high percentage of demand is served, and there is not much room left for improvement by making exchanges. In the middle ranges, where the spatiotemporal proximity between trips is moderate, is where the exchange rate is the highest. Table 8.1 lists scenarios that cover the entire range. The mean and standard deviation of the matching and exchange rates for each scenario are presented in this table. For each scenario, we have also noted the total social surplus of the system obtained due to exchanges, averaged over the 30 generated instances for each scenario.



	(500,0.1,15,9,1.5)	(200,0.25,15,9,1.5)	(500,0.9,15,9,1.5)	(200,0.75,15,9,1.5)	(200,0.5,15,9,1.5)	(200,0.6,15,9,1.5)	(500,0.7,30,9,1.5)	(500,0.7,15,25,1.5)	(500,0.7,15,9,1.5)	(1000,0.5,30,25,1.5)
(500,0.1,15,9,1.5)		0	0	0	0	0	0	0	0	0
(200,0.25,15,9,1.5)			0.002	0.002	0	0	0	0	0	0
(500,0.9,15,9,1.5)				0.533	0.006	0.027	0.022	0	0	0
(200,0.75,15,9,1.5)					0.089	0.195	0.195	0.024	0	0
(200,0.5,15,9,1.5)						0.662	0.587	0.562	0	0
(200,0.6,15,9,1.5)							0.941	0.308	0	0
(500,0.7,30,9,1.5)								0.238	0	0
(500,0.7,15,25,1.5)									0	0
(500,0.7,15,9,1.5)										0.748

Figure 8.11: P-values for the between the exchange rate is scenarios listed in table 8.1

Figure 8.11 demonstrates the statistical significance of the difference in exchange rates among scenarios. For each pair of scenarios, we have used a two-sample t-test with the null hypothesis that the two scenarios come from independent random samples drawn from two normal distributions with equal means and equal but unknown variances. The alternative hypothesis is that the two scenarios come from populations with unequal means. The null hypothesis is rejected at a 5% significance level.

Figure 8.11 shows the p-values for the two-sample t-tests for each pair of scenarios. This figure suggests that under the 5% significance level each scenario is not statistically different from one or two scenarios with higher exchange rates, but as the difference in the exchange rates between scenarios increases, the null hypothesis is rejected, and the scenarios are shown to be statistically different. This figure suggests that the difference in the exchange rates observed for different ridesharing systems with different parameter values and levels of spatiotemporal proximity between trips is statistically significant.

8.4.7 Customer Retention

The higher number of served riders that can be obtained by implementing the exchange mechanism does not have a one-to-one impact on the performance of the system. It is true that the number of served riders increases only by the number of successful exchanges, but the impact on customer retention and the reputation of the system should also be taken into consideration.

To study customer retention, we assume that a rider does not return to the system if he/she has three failed experiences. By simulating a three day experience for each rider, it is possible to compute the number of retained customers in a 3-day period. To conduct this study, we use the same set of riders in each instance of each of the scenarios listed in table 8.1 (i.e. we change only the driver set for each problem instance in each scenario). To compute customer retention, it is assumed that a rider will consider using the system again if he/she has at least one successful experience. The percentage of retained riders due to exchange are reported in table 8.1.

In addition to creating a positive experience for these riders and increasing the probability of them returning to the system, using P2P exchange could eliminate the possible negative word of mouth that could have been generated by these riders had they not been served, and could even replace them with a positive word of mouth.

8.4.8 Higher Levels of Trade

This study concentrated on the simplest possible scenario for trade, where there is a single buyer and a single seller. We study the impact of higher levels of trade on the exchange rate by considering a many-to-many ridesharing system for scenarios in Table 8.1. The results show an average of 20% increase in the number of served riders, over all scenarios. Numbers in table 8.1 are generated solely based on the operational feasibility of exchange (e.g. making sure for a rider who sells his/her current itinerary, there are alternative non-conflicting itineraries available), and not the monetary transactions. In addition, note that although theoretically a multi-level trade can increase the percentage of served riders significantly, moving to lower levels of trade can make the trade harder to manage and less probable to succeed, due to the requirement for all individuals involved in the trade to agree to it.

Chapter 9

Ride-Matching with Stochastic Demand

9.1 Introduction

In an on-demand and dynamic ridesharing system, riders may register their requests not long before they plan on starting their trips. Knowing such requests in advance can help increase the matching rate. Stochastic ride-matching attempts to achieve higher performance by using a prediction of rider arrivals, and routing drivers through paths where higher levels of demand are anticipated, although not officially registered.

In this chapter, we formulate the P2P ride-matching problem under stochastic demand for rides. In reality, riders' requests can be predicted to a good degree of accuracy using historical data, especially due to the prevalence of using smart phones to access transportation services. Predicting demand can be more straight forward during morning and evening peak hours, and in areas where there are rather distinct residential and business districts. When predicting riders' requests, the probability of each rider request r to arrive at the system can also be estimated. We denote this probability by P_r.

In order to solve the stochastic ride-matching problem, we have to generate a set of scenarios. Scenarios are all the possible combinations of the predicted riders registering in the system. The solution to the stochastic ride-matching problem determines how drivers should be re-routed under each scenario. Therefore, as the predicted riders join the system in real-time, the solution on how the drivers should be re-routed to serve them is readily available.

9.2 Generating Scenarios

We start solving the stochastic P2P ride-matching problem by generating a set of scenarios, denoted by $Scen$. Each scenario is presented as a zero/one string, with $|R_f|$ number of place holders. let $b_{i,r}$ denote the r^{th} element of scenario i. $b_{i,r} = 0$ implies that stochastic rider r is not included in scenario i (i.e. it is assumed that this rider will not register in the system), and $b_{i,r} = 1$ indicates the opposite. For example, if we predict that four riders will sign up in the future, ($|R_f| = 4$), then the string 0101 is a scenario that indicates that among the four predicted riders, only riders 2 and 4 will eventually register in the system.

Originally, the number of scenarios is exponential in $|R_f|$. In this section, we identify and eliminate scenarios that can be proven to be dominated by other scenarios, using a scenario generation algorithm. We label a scenario as "dominated" if the system cannot serve all the riders in the scenario. In our example of four riders, assume that scenario 0101 actually happens and riders 2 and 4 register in the system. If by studying the spatiotemporal proximity of the set of deterministic drivers, we come to the conclusion that rider 2 cannot find a match in the system, then whether rider 2 registers in the system or not does not impact our solution, because no matter how we route the drivers, rider 2 cannot be served. Therefore, we call scenario 0101 a dominated scenario. Since rider 2's request cannot be satisfied, scenario 0101 is in fact equivalent to scenario 0001. We call scenario 0001 the "dominant" scenario.

The probability of each scenario occurring can be calculated based on the probabilities of riders included in the scenario. Equation (9.1) demonstrates how this probability can be calculated. Needless to say, sum of the probabilities of all scenarios should be 1. Notice that calculating a scenario's probability in equation 9.1 is based on the assumption that the decisions of predicted riders whether to register in the system are (conditionally) independent of each other.

$$Prob_i = \prod_{r \in R_f | b_{i,r}=1} P_r \prod_{r \in R_f | b_{i,r}=0} (1 - P_r); \ \forall i \in Scen \tag{9.1}$$

We start the scenario generation procedure by generating an exhaustive set of scenarios, $Scen$. We denote the size of this set by N. It is easy to see that $N = 2^{|R_f|}$, since each scenario is a string with $|R_f|$ number of place holders, each of which could hold values 0 or 1. We use the notation n_i to denote the size of scenario i, i.e. sum of the digits of the string representing the scenario. We sort the scenarios in an ascending order of their size.

Each scenario could have three different states, presented in equation (9.2). Initially we set the state of all scenarios to 2. We go through the sorted list of scenarios one by one, and analyze them to determine whether a scenario is dominant and should stay in the scenario list, or is dominated, and needs to be merged with a dominant scenario.

$$State(n_s) = \begin{cases} 0 & \text{analyzed, and dominated} \\ 1 & \text{analyzed, and dominant} \\ 2 & \text{Not analyzed} \end{cases} \tag{9.2}$$

When a scenario i is labeled as dominated, there is a dominant scenario j ($n_j < n_i$) with which it should be merged. In addition, all scenarios k with $n_k > n_i$ which contain scenario i (scenarios that contain 1 in all the digits that scenario i contains 1) are also dominated scenarios. However, it is not instantaneously obvious for a dominated scenario k with which dominant scenario it has to be merged. Therefore, when finding a dominated scenario of size n, we only change the state of the scenarios of size $n+1$.

The purpose of merging a dominated scenario with its corresponding dominant scenario is to adjust the probabilities of the dominant scenarios. This is necessary since after eliminating the dominated scenarios, sum of the probabilities of all the remaining scenarios should remain 1.

Algorithm (2) shows the procedure used to generate the set of dominant scenarios in detail. This algorithm starts by generating an exhaustive set of scenarios, and setting the state of all scenarios to 2. Then the algorithm goes through the sorted list of scenarios. If a scenario k has a state of 2, then we should solve a deterministic ride-matching problem that includes the riders in the scenario, and all the registered drivers. After solving this matching problem, we keep the list of riders who can be served in set $Served(k)$. If the solution to the matching problem shows that not all riders in the scenario can be served, we change the state of the scenario to 0, and find the scenario's corresponding dominant scenario (s'). s' is a smaller scenario that includes only the riders in the current scenario that could be served. We add the tuple $[k, s']$ to set $Switch\_Scen$ to indicate that scenario k should be temporarily merged with scenario s'. This arrangement is temporary since scenario s' might itself need be merged with another scenario. Furthermore, we add to set $Not\_Served(k)$ the list of riders from scenario k that could not be served.

Next, we have to find scenarios that are one unit larger than scenario k and include all the riders in scenario k, change their status to 0, find their corresponding temporary dominant scenarios by eliminating the unserved riders in scenario k from their list of riders, and finally make a note of this probable dominated/dominant relationship in set $Switch\_Scen$. In addition, we have to add the unserved riders in scenario k to the Not_Served sets of these larger scenarios.

If after solving a scenario with initial state 2 we learn that all the riders in the scenario can be served, we simply change the status of the scenario to 1. If we come to a scenario with status of 0, we once more find scenarios that are one unit larger in size and contain the scenario, remove the unserved set of riders from the current scenario from these scenarios to find their temporary dominant scenarios, and revise the set $Switch\_Scen$ accordingly.

After we go through all the scenarios, we have to use set $Switch\_Scen$ to adjust the probabilities of dominant scenarios. We start from the last entry of this set (say $[k_1, k_2]$), and increment the probability of scenario k_1 by the probability of scenario k_2. The probability of scenario k_2 then needs to be set to zero.

9.2.1 Example

Here, we use a small example with three predicted riders to demonstrate how Algorithm (2) can be used to revise an exhaustive set of scenarios. The iterations of the scenario generation procedure are shown in Table 9.1. The changes made in each iteration are marked in red.

An exhaustive list of scenarios can be generated considering all possible combinations of the three predicted riders registering in the system. An ordered version of this list is demonstrated in Table (9.1). We start by scenario 2 (scenario 1 is always dominant). Since the status of this scenario is 2, a (deterministic) ride-matching problem needs to be solved for this scenario. The solution to this problem shows that rider 1 cannot be served, even without facing competition from any of the deterministic riders. Since scenario 2 has only one rider, the fact that this rider's request cannot be satisfied implies that we can merge scenario 2

Algorithm 2 Scenario Generation Procedure

```
Generate the exhaustive set of scenarios $Scen$ ascending in size, $n_i$.
$Done \leftarrow 0$
$n \leftarrow 0$
$k \leftarrow 1$
$State\,(k) = 2 \qquad \forall k \in Scen$
While $Done = 0$
    If $k < N$ $k \leftarrow k + 1$ Else $Done \leftarrow 1$; break End If
    $Served(k) \leftarrow$ Served riders in the optimal solution of the matching problem that includes riders in scenario $k$, and all the drivers
    If $State(k) = 2$
        If not all riders in scenario $k$ served
            $s' \leftarrow$the sub-scenario that can be served
            $s'_d \leftarrow$ the riders in scenario $k$ that cannot be served
            $n = n + 1$
            $Switch\_Scen\,(n, 1) \leftarrow k$
            $Switch\_Scen\,(n, 2) \leftarrow s'$
            $Not\_Served\,(k) \leftarrow Not\_Served\,(k) \cup \{s'_d\}$
            $State\,(k) \leftarrow 0$
            $Cand\_for\_Removal \leftarrow$ find scenarios of size $n_k + 1$ that include all the riders in scenario $k$
            For $i \in Cand\_for\_Removal$
                $State\,(cand\_for\_removal) \leftarrow 0$
                $Cand\_for\_Replacement\,(i) \leftarrow Cand\_for\_Removal\,(i) - \{s'_d\}$
                $n \leftarrow n + 1$
                $Switch\_Scen\,(n, 1) \leftarrow Cand\_for\_Removal\,(i)$;
                $Switch\_Scen\,(n, 2) \leftarrow Cand\_for\_Replacement\,(i)$
                $Not\_Served\big(Cand\_for\_Removal\,(i)\big) \leftarrow Not\_Served\big(Cand\_for\_Removal\,(i)\big) \cup \{s'_d\}$
            End For
        Else
            $State\,(k) \leftarrow 1$
        End If
    Else $State\,(k) = 0$
        $Cand\_for\_Removal \leftarrow$ find scenarios of size $n_k + 1$ that include all the riders in scenario $k$
        For $i \in Cand\_for\_Removal$
            $Cand\_for\_Replacement\,(i) \leftarrow Cand\_for\_Removal\,(i) - \{Not\_Served(i)\}$
            $n \leftarrow n + 1$
            $Switch\_Scen\,(n, 1) \leftarrow Cand\_for\_Removal\,(i)$
            $Switch\_Scen\,(n, 2) \leftarrow Cand\_for\_Replacement\,(i)$
            $Not\_Served\big(Cand\_for\_Removal\,(i)\big) \leftarrow Not\_Served\big(Cand\_for\_Removal\,(i)\big) \cup \{s'_d\}$
        End For
    End If
End While
For $i = Switch\_Scen(n, :) : Switch\_Scen(1, :)$
    $Prob\big(Switch\_Scen(i, 2)\big) \leftarrow Prob\big(Switch\_Scen(i, 2)\big) + Prob\big(Switch\_Scen(i, 1)\big)$
    $Prob\big(Switch\_Scen(i, 1) = 0$
End For
For $i \in Scen$
    If $Stat(i) = 1$,$Scen(i) = [\,]$; End If
End For
```

Table 9.1: An example of the scenario generation algorithm for a problem with three stochastic riders

	Scenario ID	1	2	3	4	5	6	7	8	
	Scenario	000	100	010	001	110	101	011	111	$Switch\_Scen$
	n_i	0	1	1	1	2	2	2	3	
Itr0	$State(i)$	2	2	2	2	2	2	2	2	$[\ ]$
	Not_Served	–	–	–	–	–	–	–	–	
Itr1	$State(i)$	2	0	2	2	0	0	2	2	$\begin{bmatrix} 2 & 1 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	–	–	$\{r_1\}$	$\{r_1\}$	–	–	
Itr2	$State(i)$	2	0	1	2	0	0	2	2	$\begin{bmatrix} 2 & 1 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	-	–	$\{r_1\}$	$\{r_1\}$	–	–	
Itr3	$State(i)$	2	0	1	1	0	0	2	2	$\begin{bmatrix} 2 & 1 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	–	–	$\{r_1\}$	$\{r_1\}$	–	–	
Itr4	$State(i)$	2	0	1	1	0	0	2	0	$\begin{bmatrix} 2 & 1 \\ 5 & 3 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	–	–	$\{r_1\}$	$\{r_1\}$	–	$\{r_1\}$	
Itr5	$State(i)$	2	0	1	1	0	0	2	0	$\begin{bmatrix} 2 & 1 \\ 5 & 3 \\ 6 & 4 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	–	–	$\{r_1\}$	$\{r_1\}$	–	$\{r_1\}$	
Itr6	$State(i)$	2	0	1	1	0	0	1	0	$\begin{bmatrix} 2 & 1 \\ 5 & 3 \\ 6 & 4 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	–	–	$\{r_1\}$	$\{r_1\}$	–	$\{r_1\}$	
Itr7	$State(i)$	2	0	1	1	0	0	1	0	$\begin{bmatrix} 2 & 1 \\ 5 & 3 \\ 6 & 4 \\ 8 & 7 \end{bmatrix}$
	Not_Served	–	$\{r_1\}$	–	–	$\{r_1\}$	$\{r_1\}$	–	$\{r_1\}$	
	$Prob_i$	$\sum\limits_{i\in\{2,1\}} Prob_i$	0	$\sum\limits_{i\in\{5,3\}} Prob_i$	$\sum\limits_{i\in\{6,4\}} Prob_i$	0	0	$\sum\limits_{i\in\{8,7\}} Prob_i$	0	

with scenario 1. This is demonstrated in the last column of Table (9.1). The first number in this column shows the current scenario, and the second number shows the dominant scenario with which the current scenario needs to be merged. In addition, since the current scenario has size of 1, we identify the scenarios with size of 2 that include rider 1 (scenarios 5 and 6), and change their status to zero. We know that these scenarios are dominated, but at this point, we do not know their corresponding dominant scenarios.

In iteration 2 we proceed to investigate scenario 3. Since the status of this scenario is 2, it should be solved. We solve the ride-matching problem, and since the only rider in this scenario (rider 2) can be matched successfully, we change the status of this scenario to 1. In the third iteration, we proceed to scenario 4 (with status 2), and solve the ride-matching problem for $R_f = \{r_3\}$. This rider can be served, and therefore similar to the previous iteration, the only thing we have to do is to change the status of this scenario to 1.

In iteration 4, we analyze scenario 5. Since the status of this scenario is already 0, there is no need to solve a ride-matching problem. We can determine which scenario it should be merged with, by studying the set $Not\_Served\,(5)$. This set contains only r_1. Eliminating r_1 from scenario 5 ($\{r_1, r_2\}$), we conclude that this scenario should be merged with scenario 3 ($\{r_2\}$). Furthermore, since $n_5 = 2$, we look for scenarios with size of 3 that contain scenario 5 (e.g. have +1 elements in digits where scenario 5 does). There is only one scenario, scenario 8, that fits this criterion. We change the status of scenario 8 to 0, and since the unserved rider in scenario 5 is r_1, we add r_1 to $Not\_Served\,(8)$.

In iteration 5, we study scenario 6. Similar to the previous scenario, this scenario does not need to be solved. Studying $Not\_Served\,(6)$ indicates that this scenario should be merged with scenario 4. Furthermore, this scenario impacts the status of scenario 8. However, we changed the status of scenario 8 to 0 in the previous iteration, and so do not need to make any additional changes to the status of this scenario. Furthermore, r_1 which is in the set $Not\_Served\,(6)$, has been already included in $Not\_Served\,(8)$, and hence no further action is required.

In iteration 6, we investigate scenario 7 which includes two riders, r_2 and r_3. Since the status of this scenario is 2, this problem needs to be solved. The solution indicates that both riders can be served. Therefore we update the status of this scenario to 1, and proceed to the last scenario. Scenario 8 has a status of zero, and studying $Not\_Served\,(8)$ suggests that it should be merged with scenario 7.

After going through the scenario list, we go through the $Switch\_Scnerio$ table, and calculate the updated probabilities of scenarios. Note that in this example instead of solving 8 matching problems for the 8 scenarios, we solved only 4 matching problems to prepare the final list of scenarios. As a general rule, eliminating a scenario with size n, reduces the number of scenarios by $2^{|R_f|-n}$. In this example, the only ride-matching problem whose all riders were not served was scenario 1. Since $n_1 = 1$, this reduced the number of scenarios to $2^3 - 2^{3-1} = 4$.

In solving the ride-matching problem associated with each scenario, instead of directly using the mathematical formulation presented in Chapter 4, it is much more efficient to use the decomposition algorithm described in Chapter 7. Since our scenarios are ordered ascending in size, once we come across a given scenario, solutions to all the sub-problems that need to be solved by the decomposition algorithm are readily available. This makes solving the deterministic matching problems in Algorithm 2 exceptionally quick.

In our example in this section, for instance, in scenario 7, we have riders r_2 and r_3. We already have the separate solutions for the $\{r_2\}$ and $\{r_3\}$ sub-problems, by solving scenarios 3 and 4. If the itineraries of these riders can co-exist (i.e. they are matched with different drivers, or with the same set of drivers but through the same routes for the drivers), then the solution to scenario 7 is readily available. Using this decomposition algorithm proves much more important when dealing with a larger list of scenarios, since scenarios of size n

all include scenarios of size $n-1$. Assuming that the solutions to scenarios of size $n-1$ are available, using this decomposition algorithm, not much computation is required for solving scenarios of size n. In this chapter, we use this decomposition algorithm to generate the list of dominant scenarios much more efficiently.

9.3 P2P Multi-Hop Ride-Matching Problem with Stochastic Demand

The P2P ride-matching problem with stochastic demand can be formulated as a stochastic program with fixed recourse. The first stage decision variables determine the itineraries of drivers and deterministic riders. The second stage variables are scenario-dependent, and determine the itineraries of the stochastic riders and drivers under each scenario. These variables are presented in equations (9.3)-(9.6).

$$x'^{d}_{\ell k} = \begin{cases} 1 & \text{Driver } d \text{ travels on link } \ell \text{ in scenario } k \\ 0 & \text{Otherwise} \end{cases} \tag{9.3}$$

$$y'^{rd}_{\ell k} = \begin{cases} 1 & \text{Driver } d \text{ carries rider } r \text{ on link } \ell \text{ in scenario } k \\ 0 & \text{Otherwise} \end{cases} \tag{9.4}$$

$$u'^{d}_{rk} = \begin{cases} 1 & \text{Driver } d \text{ carries rider } r \text{ in scenario } k \\ 0 & \text{Otherwise} \end{cases} \tag{9.5}$$

$$z'_{r,k} = \begin{cases} 1 & \text{Rider } r \text{ is matched in scenario } k \\ 0 & \text{Otherwise} \end{cases} \tag{9.6}$$

In the interest of simplicity of notation, let us define set $Scen' = \{Scen \cup 0\}$ as the union of scenario 0 and our former scenario set $Scen$. We define scenario 0 to include the deterministic riders, and set $R_t = \{R \cup R_f\}$ to include the entire set of riders, deterministic and stochastic. In addition, we define a binary parameter $b_{r,k}$ to be valued 1 if rider r is part of scenario k, and 0 otherwise. Needless to say, $b_{r,0} = 1, \forall r \in R$ (i.e. scenario 0 contain the set of deterministic riders). In addition, all the deterministic riders are included in the formulation with probability of 1.

Since the stochastic ride-matching program with recourse is only piecewise linear, we reformulate it as a binary (i.e. zero/one) program, and solve this deterministic equivalent instead. The set of constraints for this binary reformulation, P, is presented in model (9.7). Constraint sets (9.7b)-(9.7e) route drivers in the network, and ensure that they do not exceed their travel time windows and maximum ride times. Constraint sets (9.7f)-(9.7l) route riders, making sure that they do not exceed their travel time windows, maximum ride times,and maximum number of transfers.

Constraint set (9.7m) ensures that parts of the drivers' routes that pertains to transporting the deterministic riders do not change under different scenarios, since in the beginning of each re-optimization period the deterministic riders' routes are announced to them and cannot be changed. Constraint set (9.7n) links the stochastic riders to the deterministic riders and drivers, by making sure that the vehicle capacities are not exceeded under any scenarios. Finally, constraint set (9.7o) determines which riders belong to which scenarios.

The objective function of the problem presented in equation(9.7a) maximizes the expected objective function of the problem. Any of the terms introduced in Chapter 4 can be used as the objective function. Recall that we use probabilities of 1 for the deterministic riders.

$$P : \text{Maximize} \quad Z_{sto} = Z + \sum_{k \in Scen} Prob_k Z_k \tag{9.7a}$$

$$\sum_{\substack{\ell \in L: \\ s_i = OS_d}} x'^{d}_{\ell,k} - \sum_{\substack{\ell \in L: \\ s_j = OS_d}} x'^{d}_{\ell,k} = 1; \; \forall k \in Scen', \forall d \in D \tag{9.7b}$$

$$\sum_{\substack{\ell \in L: \\ s_j = DS_d}} x'^{d}_{\ell,k} - \sum_{\substack{\ell \in L: \\ s_i = DS_d}} x'^{d}_{\ell,k} = 1; \; \forall k \in Scen', \forall d \in D \tag{9.7c}$$

$$\sum_{\substack{t_i, s_i \\ \ell = (t_i, s_i, t, s) \in L}} x'^{d}_{\ell,k} = \sum_{\substack{t, s_j \\ \ell = (t, s, t_j, s_j) \in L}} x'^{d}_{\ell,k}; \; \forall k \in Scen', \forall d \in D, \forall t \in T_d, \forall s \in S \backslash \{OS_d \cup DS_d\} \tag{9.7d}$$

$$\sum_{\ell \in L} (t_j - t_i) x'^{d}_{\ell,k} \leq T^{TB}_d; \; \forall k \in Scen, \forall d \in D \tag{9.7e}$$

$$\sum_{d \in D'} \sum_{\substack{\ell \in L: \\ s_i = OS_r}} y'^{rd}_{\ell,k} - \sum_{d \in D'} \sum_{\substack{\ell \in L: \\ s_j = OS_r}} y'^{rd}_{\ell,k} = z'_{r,k}; \; \forall k \in Scen', \forall r \in R_t \tag{9.7f}$$

$$\sum_{d \in D'} \sum_{\substack{\ell \in L: \\ s_j = DS_r}} y'^{rd}_{\ell,k} - \sum_{d \in D'} \sum_{\substack{\ell \in L: \\ s_i = DS_r}} y'^{rd}_{\ell,k} = z'_{r,k}; \; \forall k \in Scen, \forall r \in R_t \tag{9.7g}$$

$$\sum_{d \in D'} \sum_{\substack{t, s_i: \\ \ell = (t_i, s_i, t, s) \in L}} y'^{rd}_{\ell,k} = \sum_{d \in D'} \sum_{\substack{t_j, s_j: \\ \ell = (t, s, t_j, s_j) \in L}} y'^{rd}_{\ell,k}; \; \forall k \in Scen', \forall r \in R_t, \forall t \in T_r, \forall s \in S \backslash \{OS_r \cup DS_r\} \tag{9.7h}$$

$$\sum_{d \in D'} \sum_{\ell \in L} (t_j - t_i) y'^{rd}_{\ell,k} \leq T^{TB}_r; \; \forall k \in Scen', \forall r \in R_t \tag{9.7i}$$

$$u'^{d}_{r,k} \geq y'^{rd}_{\ell,k}; \; \forall k \in Scen', \forall r \in R_t, \forall d \in D, \forall \ell \in L \tag{9.7j}$$

$$u'^{d}_{r,k} \leq \sum_{\ell \in L} y'^{rd}_{\ell,k}; \; \forall k \in Scen', \forall r \in R_t, \forall d \in D \tag{9.7k}$$

$$\sum_{d \in D} u'^{d}_{r,k} - 1 \leq V_r; \; \forall r \in R_t \tag{9.7l}$$

$$x'^{d}_{\ell,k} \geq y'^{rd}_{\ell,0}; \; \forall k \in Scen, \forall r \in R, \forall d \in D, \ell \in L \tag{9.7m}$$

$$\sum_{r \in R_t} y'^{rd}_{l,k} \leq C_d x'^{d}_{\ell,k}; \; \forall k \in Scen', \forall d \in D, \forall \ell \in L \tag{9.7n}$$

$$z'_{r,k} \leq b_{r,k}; \; \forall r \in R_t, \forall k \in Scen' \tag{9.7o}$$

9.4 L-Shaped Decomposition

In this section we use L-shaped decomposition to solve the deterministic equivalent of the ride-matching problem in model (9.7) more efficiently from a computational point of view. An L-shaped decomposition iteratively solves a master problem containing a subset of constraints of the original problem and $|Scen|$ number of sub-problems, until a certain criterion is met. In each iteration, the solution of the master problem is fixed in the iteration's sub-problems, making it possible for the sub-problems to be solved independently, and in parallel.

We formulate the master problem to route the deterministic riders and the drivers. We then formulate the sub-problems to route the stochastic riders and re-route the drivers while fixing the portions of their itineraries that are allocated to the deterministic riders. After solving the master problem in each iteration, we fix the itineraries of the drivers and the deterministic riders. This makes the sub-problems independent of each other.

The master problem is a relaxation of the original problem. In each iteration, after solving the master problem, we generate optimality cuts based on the feedback received from the sub-problems (constraint set (9.8b)), and add these cuts to the master problem, shrinking the feasible region.

We start by dividing the set of constraints in model (9.7) between the master problem and sub-problems. The master problem includes the set of constraints for drivers and deterministic riders (e.g. constraint sets in model 9.7 that pertain to scenario 0). In addition, the master problem includes a set of optimality cuts that are presented in constraint set (9.8b). In addition to the binary variables presented in (9.8e), there is an additional variable, θ, in the set of decision variables of the master problem. Variable θ is a lower bound on sum of the sub-problems' objective function values. Vectors $b_{k,i}$ and $u_{k,i}$ in constraint set (9.8b) are the right hand side and dual values of the constraint sets of sub-problem k in iteration i, respectively. Since the decision variables in the sub-problems are relaxed, the internal product of vectors b and u for a sub-problem renders the sub-problem's optimal objective value (strong duality). The optimality cuts generated raise the lower bound θ until the stopping criteria of $\theta = \sum_{k \in Scen} Prob_k \, Z_{SP}(i,k)$ is reached in some iteration i, where $Z_{SP}(i,k)$ is the objective function value of sub-problem k in iteration i. The vector b includes the master problem variables that are fixed in the sub-problems, and so $b \cdot u$ is a constant that helps us evaluate the stopping criterion. However, if the stopping criterion is not met, and when we are generating the optimality cuts, these formerly fixed master problem variables are treated as decision variables in the right hand side of constraint set (9.8b).

Master Problem (i) $\quad i \in \{i..itr\}$:

$$\text{Minimize} \quad Z_{MP}(i) : \, Z + \theta \tag{9.8a}$$

$$\theta \geq \sum_{k \in Scen} Prob_k \times \big(b_k u_{k,i}\big); \; \forall i \in \{2..itr\} \tag{9.8b}$$

$$\theta \geq 0 \tag{9.8c}$$

$$P\big((9.7b) - (9.7l), (9.7n) - (9.7o), k = 0\big) \tag{9.8d}$$

$$x'^{d}_{\ell,0}, y'^{rd}_{\ell,0}, u'^{d}_{r,0}, z'_{r,0} \in \{0,1\} \tag{9.8e}$$

Sub-problems include the set of constraints pertaining to stochastic riders. As mentioned before, the itineraries of the drivers and the deterministic riders are fixed in sub-problems. This makes it possible to dedicate each sub-problem to a separate scenario (model (9.9)).

The variables that were binary previously are relaxed in the sub-problems. We add constraint set (9.9c) to model (9.9) to ensure that only the remaining capacity of vehicles (taking into consideration the capacity consumed by the deterministic riders) is being used by sub-problems. This helps reduce the number of iterations needed to reach the stopping criterion. Furthermore, there is no source of infeasibility in the sub-problems, and therefore sub-problems only generate and send an optimality cut per iteration to the master problem.

Sub-problem $(i,k) \qquad \forall i \in 1..itr, \forall k \in Scen$:

$$\text{Minimize} \quad Z_{SP}(i,k):\ Z(k) \tag{9.9a}$$

$$P\big((9.7b)-(9.7o),k\big) \tag{9.9b}$$

$$\sum_{r\in R_f} y'^{rd}_{l,k} + \sum_{r\in R} y'^{rd}_{l,0} \le C_d x'^{d}_{l,k};\ \forall k \in Scen', \forall d \in D, \forall l \in L \tag{9.9c}$$

$$0 \le x'^{d}_{\ell,0}, y'^{rd}_{\ell,0}, u'^{d}_{r,0}, z'_{r,0} \le 1 \tag{9.9d}$$

After the stopping criterion is met, we solve the last iteration's sub-problems one additional time, with (9.9d) replaced by integrality constraints. Note that the sub-problems are always feasible, and hence enforcing the decision variables to be binary does not lead to an infeasible solution to the original problem.

The sub-problems, in general, have very tight linear relaxations. In more than 90% of the problem instances we solved, the solutions to the linear relaxations turned out to be integer. For the remaining 10%, enforcing the binary condition led to an objective function equal to the integer part of the objective function of the linear relaxation of the problem, implying that the solutions remained optimal. We have never encountered an instance of the problem in which enforcing the binary constraints has led to sub-optimal solutions.

9.5 Small Example

In this section, we present a small example of a ridesharing system with two registered (deterministic) riders, three stochastic (predicted) riders, and two drivers, and solve the problem and present the solution using both the deterministic reformulation of the stochastic program with recourse presented in (9.7), and the L-shaped decomposition. In this example, nine stations are located in the network, where riders can start their trips, end them, or

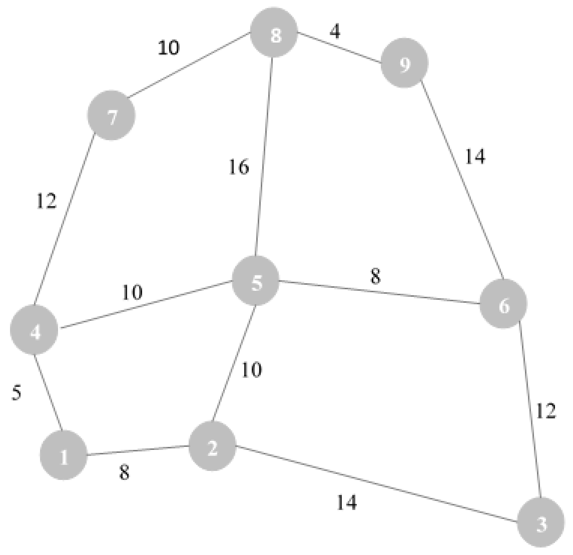


Figure 9.1: Stations and travel times in the example network

Table 9.2: Input data for the example

	Deterministic Riders		Stochastic Riders			Drivers		
Participant	r_1	r_2	r_3	r_4	r_5	d_6	d_7	d_8
Rider(0)/Driver(1)	0	0	0	0	0	1	1	1
Origin Station	3	4	5	2	9	3	9	6
Destination Station	2	6	2	8	7	7	8	8
Earliest Departure	5	5	6	20	41	5	41	4
Latest Arrival	21	24	16	46	56	56	46	23
Probability	1	1	0.7	0.9	0.85	1	1	1

make transfers between drivers. Figure 9.1 displays the network for this example. Table 9.2 shows the problem instance we are trying to solve in this example. We use the objective of maximizing the number of served riders.

The first step is to form the set of scenarios. The step by step scenario generation procedure presented in Table 9.1 is in fact for the example in this section. The results suggest that the *Scen* set contains four scenarios. These scenarios and their corresponding probabilities are presented in Table 9.3. Note that the first scenario (all-zero string) can always be dropped from the set of scenarios.

First, we use the deterministic reformulation of the stochastic ride-matching problem with

Table 9.3: Set of scenarios with their corresponding probabilities

Scenario	Probability
000	0.0150
010	0.1350
001	0.0850
011	0.7650
sum	1.0000

Table 9.4: Optimal solution to the stochastic ride-matching problem

		Scenario	Participant	Matched?	Driver	Route (t_i, s_i, t_j, s_j)
First Stage	Matched Drivers	NA	d_6	Yes	d_6	$(6, 3, 20, 2)$
		NA				$(20, 2, 30, 5)$
		NA				$(30, 5, 46, 8)$
		NA				$(46, 8, 56, 7)$
		NA	d_7	Yes	d_7	$(42, 9, 46, 8)$
	Deterministic Riders	NA	r_1	Yes	d_6	$(6, 3, 20, 2)$
		NA	r_2	No	$-$	$-$
Second Stage Variables	Stochastic Riders	1	r_4	Yes	d_6	$(20, 2, 30, 5)$
					d_6	$(30, 5, 46, 8)$
		2	r_5	Yes	d_7	$(42, 9, 46, 8)$
					d_6	$(46, 8, 56, 7)$
		3	r_4	Yes	d_6	$(20, 2, 30, 5)$
					d_6	$(30, 5, 46, 8)$
			r_5	Yes	d_7	$(42, 9, 46, 8)$
					d_6	$(46, 8, 56, 7)$

recourse, presented in (9.7), to solve this problem. The optimal solution is displayed in Table 9.4. Based on this solution, only one of the deterministic riders can be served. However, two of the three stochastic riders can be matched in the system. The time required to generate the scenarios was less than 2 seconds. The optimal solution to the stochastic ride-matching problem was obtained in less than 0.5 second.

Next, we use the L-shaped decomposition to solve the problem. The final solution is exactly the same as the solution in Table 9.4 (The solution to the last sub-problem was integer). It took six iterations for the algorithm to converge. The evolution of variable θ is presented in Figure 9.2. As expected, the value of θ is increasing through iterations, since we keep adding

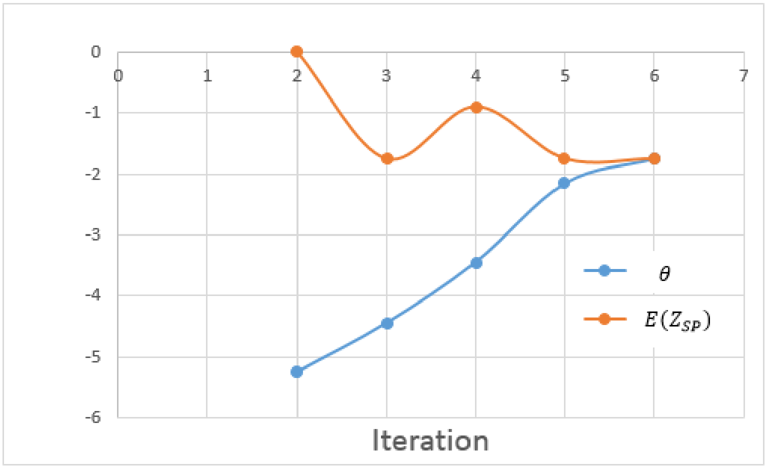


Figure 9.2: Evolution of θ over iterations of L-shaped decomposition, and convergence of the L-shaped decomposition

Table 9.5: Solution to the deterministic ride-matching problem

	Participant	Matched?	Driver	Route (t_i, s_i, t_j, s_j)
Deterministic Riders	r_1	Yes	d_6	$(7, 3, 21, 2)$
	r_2	No	$-$	$-$
Matched Drivers	d_6	Yes	d_6	$(7, 3, 21, 2)$
				$(21, 2, 29, 1)$
				$(29, 1, 34, 4)$
				$(34, 4, 46, 7)$

optimality cuts to the master problem.

In deterministic ridesharing systems where riders' requests are not predicted, the ride-matching problem is initially solved only for the deterministic (registered) riders. Once (and if) a rider joins the system in real-time, a ride-matching problem can be solved for the newly joined rider assuming fixed routes for drivers who have been assigned to deterministic riders. We use this small example to show the superiority (in terms of system performance) of predicting rider requests and solving a stochastic program, to real-time matching of individuals.

To this end, we first solve a matching problem for the set of deterministic riders, $R = \{r_1, r_2\}$, using the decomposition algorithm in Chapter 7. The solution to this problem is displayed in

Table 9.5. This solution indicates that from the set of deterministic riders, $R = \{r_1, r_2\}$, only r_1 can be served. d_6 is the driver who is matched with r_1. However, notice that d_6's itinerary is different from his itinerary in Table 9.4. Next, we assume that the three stochastic riders (whose arrival was not predicted) arrive in real-time. We use the DP algorithm described in Chapter 6 to find an itinerary for them. In forming the time-expanded feasible networks for these riders, we assume that the itinerary of d_6 is fixed, but d_7 and d_8 can be routed, since they were not matched previously. The solutions show that none of the real-time (stochastic) riders can find a match in the system. This example illustrates the importance of formulating and solving a stochastic ride-matching problem. Although in both cases (the stochastic, and deterministic problems) only r_1 from the set of deterministic riders could be served, in the stochastic case, we could serve riders r_4 and r_5 as well.

9.6 Applicability in Practice

In this section, we investigate the possibility of using stochastic matching for problems of practical sizes. To this end, we have generated multiple random instances of the ride-matching problem, and have recorded their solution times, and the quality of their solutions. The problem instances are generated on a random greed-like network similar to the one presented in Figure 9.1. We vary the number of drivers, and deterministic and stochastic riders in each instance. The experiments are performed on a PC with Core i7 3 GHz and 8GB of RAM.

It should be noted that a ride-sharing system could be very sparse in time and space, and therefore for each problem instance, we first use the pre-processing procedure ESTAM presented in Chapter 5 to detect and filter out a subset of riders who cannot be matched. The terms "before" and "after" under rows $|R|/|R_f|$ in table 9.6 refer to the number of deterministic/stochastic riders, before and after performing the pre-processing procedure, respectively.

The row $|Scen|$ this table shows the number of scenarios generated using Algorithm 2. We show the time of reaching the optimal solution by directly solving the binary reformulation of the problem presented in model 9.7. In addition, we report the solution time by the L-shaped decomposition, and comment on the optimality of the solution. It should be noted that since the sub-problems in the L-shaped decomposition can be solved in parallel, the time we report is the maximum solution time of sub-problems in each iteration, and not the sum of solution times. Table 9.6 suggests that larger instances of the problem can still be solved in a matter of a few minutes.

Table 9.6: Experiments with problems of larger sizes

Problem instances		1	2	3	4	5	6	7	8
$\|S\|$		16	16	25	25	49	49	64	64
$\|D\|$		30	50	60	60	90	100	110	120
$\|R\|$	Before pre-processing	15	45	40	45	50	60	60	50
	After pre-processing	4	19	13	15	11	10	11	12
$\|R_f\|$	Before pre-processing	5	25	20	20	20	25	20	30
	After pre-processing	3	8	4	8	5	4	5	6
$\|Scen\|$		7	127	7	63	31	15	15	15
Pre-processing time (s)		1	5	8	12	25	31	47	45
Scenario generation time (s)		8	8	4	15	13	15	40	28
Binary reformulation solution time (s)		1	48	4	74	92	49	299	168
L-shaped decomposition	Time	6	152	4	189	45	146	187	520
	Iterations	12	24	3	29	9	24	19	37
	Optimality	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Objective function value	Stochastic	−6.26	−20.24	−11.73	−10.92	−10.36	−10.96	−8.60	−12.87
	Expected value	−4.6	−20.24	−11.73	−10.92	−9.36	−10.96	−6.60	−9.32

Comparing the solution times of the binary reformulation of the stochastic program and the L-shaped decomposition suggests that often times the L-shaped decomposition takes longer to converge, although it is not the case in 100% of the instances. One benefit of using the L-shaped decomposition in time-sensitive applications is that since the master- and sub-problem solutions are always feasible, one can allow the iterations to run until a time limit is reached, and then retrieve the best solution. Another advantage is the possibility of distributing computations, in this case in users' devices.

Another important point is that although the integral constraints in the sub-problems of the L-shaped decomposition have been relaxed, the solutions obtained remain optimal in all instances.

In order to assess the value of the stochastic solution, we compare the solution to the stochastic program to the solution obtained from solving a deterministic ride-matching problem (model (4.5)), with the objective function of maximizing the expected value of number of served riders. The comparisons are displayed in the last two rows of table 9.6. Results suggest that in some instances, the solutions of the stochastic and deterministic problems are exactly the same, while in some others the stochastic solution is superior by up to 38%.

Chapter 10

Shared Ownership and Ridership of Autonomous Vehicles

10.1 Introduction

Sharing economy, also known as collaborative consumption, is a fairly old concept that focuses on the benefits obtained from sharing resources (products or services) that would otherwise go unused. Although communities have been using the concept of sharing economy locally for many years, advent of Internet has led to its spread in global populations, and highlighted its benefits.

The sharing economy model has been historically used for high-value commodities, such as exotic automobiles, yachts, private jets, vacations homes, and the like. Curvy Road, for example, an exotic carshare company founded in 2000, provides fractional ownership of high-end vehicles in four cities in the US (Curvy Road (2000)). Although it has long been realized that taking ownership of under-utilized high-value assets may not be always economically wise, this economic model has become more popular recently for less expensive resources as well, thanks to new platforms that allow easy and quick development of companion mobile applications.

Autonomous (also known as driver-less and self-driving) vehicles are expected to enter the market in the near future. Although these vehicles introduce many benefits such as a higher degree of safety and mobility to the users and the transportation system in general, their high prices can be prohibitive when it comes to purchasing them. On the other hand, autonomous vehicles can decrease the total number of vehicles needed to perform daily tasks, since these vehicles can drive themselves to locations where there is demand for transportation. One possible strategy to make autonomous vehicles more affordable is to encourage shared ownership of these vehicles. In addition to shared ownership, it is possible to decrease the ownership cost of autonomous vehicles further by (1) shared use of these vehicles, and (2) renting out the vehicles when they are not being used by owners.

Since autonomous vehicles can drive themselves, owning these vehicles decreases the number of vehicles a household requires to perform daily tasks. Figure 10.1 shows the average daily vehicle miles traveled (VMT) by each vehicle in a household in the US in 2009. This figure suggests that the higher the number of vehicles owned by a household, the less the average use of each additional vehicle tends to be. Although for a typical household owning more than one vehicle might be financially justifiable considering the level of comfort and peace of mind it might bring, this justifiability decreases with the purchase of additional vehicles. Apart from the initial investment (or monthly payments), the cost of insurance, depreciation of value, and parking can turn vehicle ownership into a financial burden. With autonomous vehicles, fewer vehicles can cover the same trips compared to a higher number of ordinary vehicles.

In this chapter, we introduce the shared vehicle ownership and use (SVOU) program in which a group of households jointly own and use a set of autonomous vehicles. These households can share rides with each other if the spatiotemporal distribution of their trips allow that.

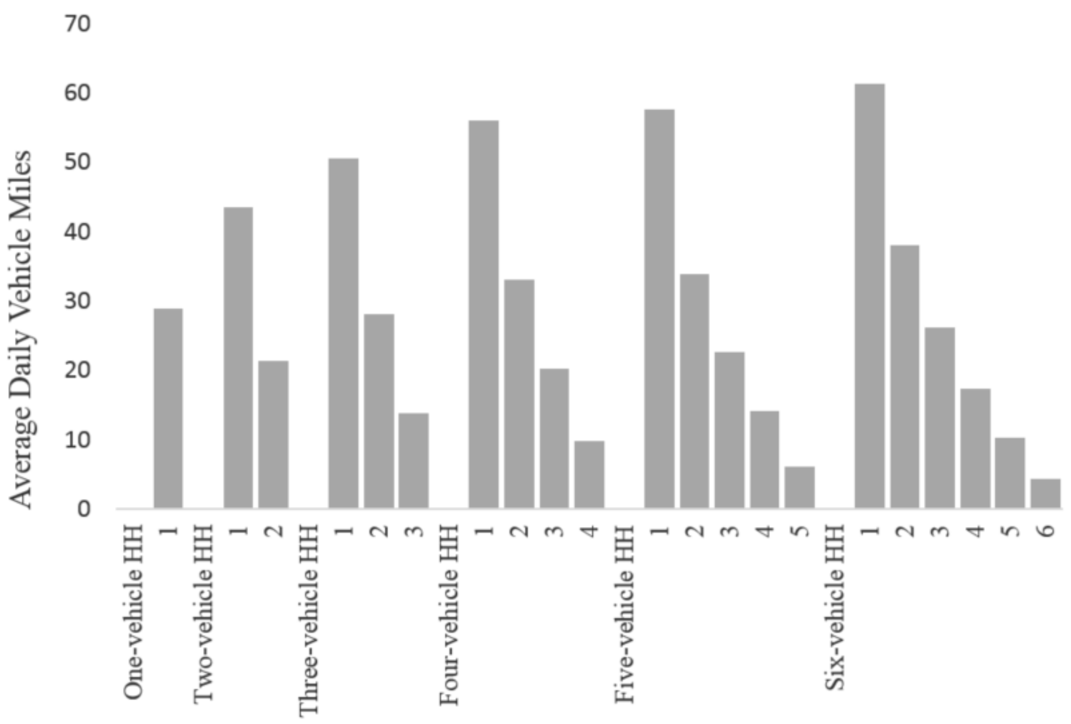


Figure 10.1: Average daily vehicle miles for households with various number of vehicles. The data is from the National Household Travel Survey (NHTS), 2009

We propose analytical optimization schemes to study the impact of this program on an individual and system level. Though no rigorous analysis under optimal operations appear to have been done in the literature yet, there is some awareness of such possibilities in the automotive industry as well as among the increasing number of researchers and aficionados of autonomous vehicles (Schall, 2015; k. Naughton, 2015; Fagnant and Kockelman, 2014; Schoettle and Sivak, 2015). In this chapter, along with an analytical formulation, we also offer certain new possibilities such as renting the vehicles out through a central car rental service when they are not being used by their owners.

The analysis presented here are equally applicable to both fully autonomous vehicles and driverless vehicles (which may not be technically autonomous), the distinction between the two terms being relatively well-known now. In the following discussions, however, we use only the term “autonomous” to avoid unnecessary repetition of the fact that it refers to driverless vehicles as well.

10.2 SVOU: Shared Vehicle Ownership and Use Program

Envision a set of households F, who share the ownership of a set of autonomous vehicles V. These households form a cluster to which the vehicles under their shared ownership belong. Each vehicle $v \in V$ has the capacity to carry C_v number of passengers. Each household $f \in F$ has a set of essential trips that need to be served by the set of autonomous vehicles. We define set M_e to include all the essential trips of the households that belong to a cluster. Common types of essential trips may include work-based trips, grocery shopping, and trips to school. However, households can include any type of trips in the set of essential trips, for which they need to ensure regular and timely access to a vehicle.

For a given trip k, a cluster member needs to input into the system the location of the origin of the trip, OS_k, the location of the destination of the trip, DS_k, the earliest departure time from the origin location, ED_k, and the latest arrival time at the destination location, LA_k. While vehicles are idle, they can be rented out to satisfy a set of on-demand transportation requests, M, in order to cover a part of the system cost. A rental request $k \in M$ should include the location where a vehicle needs to deliver itself (OS_k), and the location where it will be returned (DS_k), along with the rental period duration (P_k), and the rental time window ($[ED_k, LA_k]$).

The first goal of the system is to advise households in a cluster on the optimal number of vehicles they need to purchase to cover their set of essential trips. In the interest of higher efficiency, the system is designed to allow cluster members to rideshare, if the spatiotemporal proximity of their trips permit it. The second goal of the system is to maximize the total number of on-demand car rental requests, in order to maximize the external revenue generated. These goals are implemented sequentially, i.e. we first determine the optimal number of vehicles for each cluster of households, and then use these vehicles for carsharing during their idle times. In the next section, we mathematically model these two problems.

10.2.1 Mathematical Modeling

In order to model the system defined in the previous section, we formulate two optimization problems. The first problem finds the optimal number of autonomous vehicles that should be owned by a cluster, in order to guarantee that its set of essential trips will be served, and provides vehicle itineraries. The second problem uses the vehicles' idle times to serve the maximum number of on-demand car rental requests.

To formulate these two problems, we need to first define a number of sets. For a given cluster, we define a set of stations, S_e, that contains the origin and destination locations of the cluster's essential trips. Furthermore, we define set S to contain all the origin and destination locations of all essential and non-essential trips (by all clusters). Similar to Chapter 4, by introducing stations, we discretize the space dimension of the problem. In addition, we discretize the study time horizon into a set of short time intervals with length Δt. We define set T to include all time intervals in the study time horizon. In this chapter, we use $\Delta t = 5$ minutes. In a network discretized in both time and space, we define a node n as a tuple $(t_i, s_i) \in T \times S$. Consequently, we define a link ℓ as a tuple of nodes $\ell = (n_i, n_j) = (t_i, s_i, t_j, s_j)$, where $(t_j - t_i)\Delta t$ is the travel time between stations s_i and s_j. We define set L to include all links.

Furthermore, we define an origin depot, D_o, and a destination depot D_d. The depot stations are not real locations on the network, and are used to assist in the formulation of the problem. D_o is connected to all stations in set S, and all stations in S are connected to D_d. Furthermore, D_o and D_d are connected to each other. Figure 10.2 displays a typical network and demonstrates the connection between the depots and set of stations.

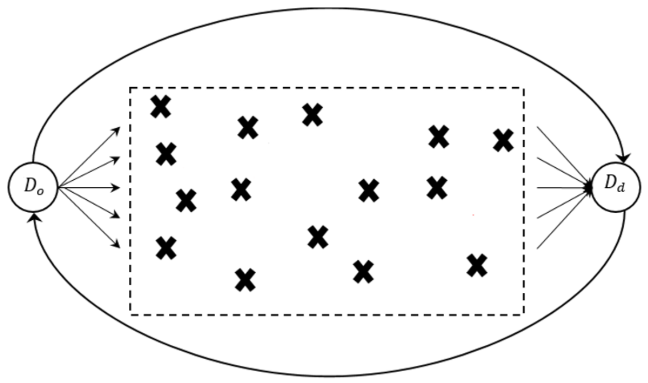


Figure 10.2: A typical network to demonstrate the connection of depot stations to each other, and to members of set S

We use the pre-processing procedure in Chapter 5 to reduce the size of the input sets to the problem. Through this procedure (the first two steps of ESTAM), we find the links that are spatiotemporally reachable for each trip $k \in \{M \cup M_e\}$ given its time window, and keep such links in set L_k. Therefore, when formulating the problem, we do not need to place explicit constraints on the time windows of trips, since only links with feasible time windows are members of set L_k.

10.2.2 Routing of Autonomous Vehicles

The problem of finding the optimal number of vehicles to serve a cluster's set of essential trips is formulated in model (10.3). The formulation requires two sets of decision variables defined in (10.1) and (10.2).

$$x_\ell^v = \begin{cases} 1 & \text{If vehicle } v \text{ travels on link } \ell \\ 0 & \text{Otherwise} \end{cases} \tag{10.1}$$

$$y_\ell^{kv} = \begin{cases} 1 & \text{If trip } k \text{ is carried out by vehicle } v \text{ on link } \ell \\ 0 & \text{Otherwise} \end{cases} \tag{10.2}$$

Given that the households in a cluster have a total of m members, $|m|$ is an upper-bound on the number of vehicles needed to serve the cluster. Therefore, to determine the minimum number of vehicles required for a given cluster, we formulate an optimization problem, assuming there to be $|m|$ vehicles available, and try to maximize the number of vehicles that are not used.

Constraint sets (10.3b) and (10.3c) force all vehicles to go back from D_d to D_o at the end of the day. Constraint set (10.3d) is the flow conservation constraint, forcing all vehicles that enter a station at a given time interval to exit that station at the same time interval. Notice that vehicles do not have to physically leave a station. Members of set L in the form $\ell = (t, s, t+1, s)$ can cover such situations, where a vehicle can stay at a station for one time interval. Constraint sets (10.3e)-(10.3g) route the set of trips in the network. Constraint set (10.3e) and (10.3f) ensure that a trip leaves its origin station and enters its destination station within the trip's time window, respectively. Constraint set (10.3g) is the flow conservation constraint. Constraint set (10.3h) serves two purposes: it links vehicle routes to trip routes, and ensures that the number of individuals assigned to each vehicle at any moment in time does not exceed the vehicle's capacity.

Vehicles that are excessive and are not actually routed in the system have to take the link that connects D_o to D_d. Therefore, to minimize the number of used vehicles, we maximize the vehicles that travel on this link, as mathematically stated in the objective function of the problem in (10.3). The second term in the objective function minimizes the total travel time by vehicles in the network. We set a negative weight W for the first term in the objective function to take into account the relative importance of minimizing the number of vehicles in a cluster, and the total travel time by the cluster members.

The solution to this problem simultaneously provides the minimum number of vehicles required to serve the essential trips, and itineraries for the trips and the vehicles.

$$\text{Minimize} \qquad W \sum_{\substack{v\in V,\, t_i\in T,\, t_j\in T:\\ \ell=(t_i,D_o,t_j,D_d)\in L}} x_\ell^v + \sum_{\substack{v\in V,\ell\in L:\\ s_i\neq s_j}} x_\ell^v \tag{10.3a}$$

$$\sum_{\substack{\ell=(t_i,s_i,t_j,s_j)\in L:\\ s_i=D_d,s_j=S_e\backslash D_o}} x_\ell^v = 0;\ \forall v\in V \tag{10.3b}$$

$$\sum_{\substack{\ell=(t_i,s_i,t_j,s_j)\in L:\\ s_i=D_d,s_j=D_o}} x_\ell^v = 1;\ \forall v\in V \tag{10.3c}$$

$$\sum_{\substack{s_i\in S_e,t_i\in T:\\ \ell=(t_i,s_i,t,s)\in L}} x_\ell^v = \sum_{\substack{s_j\in S_e,t_j\in T:\\ \ell=(t,s,t_j,s_j)\in L}} x_\ell^v;\ \forall v\in V, t\in T, s\in S_e\backslash D_o : \ell=(t_i,s_i,t,s)\in L \tag{10.3d}$$

$$\sum_{v\in V}\sum_{\substack{\ell\in L_k:\\ s_i=OS_k}} y_\ell^{kv} - \sum_{v\in V}\sum_{\substack{\ell\in L_k:\\ s_j=OS_k}} y_\ell^{kv} = 1;\ \forall k\in M_e \tag{10.3e}$$

$$\sum_{v\in V}\sum_{\substack{\ell\in L_k:\\ s_j=DS_k}} y_\ell^{kv} - \sum_{v\in V}\sum_{\substack{\ell\in L_k:\\ s_i=DS_k}} y_\ell^{kv} = 1;\ \forall k\in M_e \tag{10.3f}$$

$$\sum_{\substack{s_i\in S_e,t_i\in T:\\ \ell=(t_i,s_i,t,s)\in L_k}} y_\ell^{kv} = \sum_{\substack{s_j\in S_e,t_j\in T:\\ \ell=(t,s,t_j,s_j)\in L_k}} y_\ell^{kv};\ \forall v\in V, k\in M_e, t\in T, s\in S_e\backslash\{OS_k\cup DS_k\} : \ell=(t_i,s_i,t,s)\in L_k \tag{10.3g}$$

$$\sum_{k\in M_e,\ell\in L_k} x_\ell^{kv} \ \leq\ x_\ell^v;\ \forall v\in V, \ell\in L \tag{10.3h}$$

10.2.3 On-Demand Carsharing

The pick-up and drop-off schedules for the set of essential trips are determined by the optimization problem in the previous section. The idle times of vehicles can be used to serve on-demand transportation requests through a central car rental service provider.

Autonomous vehicles become free after dropping off a passenger, and before picking up the next. During this period, a vehicle needs to make a trip from the destination location of the first passenger, to the origin location of the second, in a travel time window that is bounded from below by the scheduled arrival time of the first passenger, and from above by the scheduled departure time of the second. The first optimization problem ensures that this travel time window is larger than the actual travel time between the locations. Although this time window is not strictly an idle period (the vehicle should travel to the pick-up location for the second trip), the vehicle can use the extra time to serve on-demand car rental requests, and that is why we refer to the time window between two consecutive scheduled drop-off and pick-ups as the free travel time window.

In order to mathematically formulate the carsharing problem, we identify the set of free time windows between scheduled trips, and try to find the maximum number of carsharing requests that can be satisfied during these time windows. We keep the set of free time windows for each vehicle $v \in V$ in set $J(v)$. The j^{th} free time window of autonomous vehicle v starts after dropping off its j^{th} assigned passenger, and ends when passenger $j+1$ needs to be picked up. We denote this travel time window by $[ED_{(v,j)}\ LA_{(v,j)}]$. During this time window, the vehicle needs to travel from the destination location of its j^{th} scheduled trip, to the origin location of its next scheduled trips. We denote these parameters by $OS_{(v,j)}$ and $DS_{(v,j)}$ respectively, and formulate this problem using three sets of decision variables in equations (10.4)-(10.6).

$$x_\ell^{vj} = \begin{cases} 1 & \text{If vehicle } v \text{ travels on link } \ell \text{ during its } j^{th} \text{ free time window} \\ 0 & \text{Otherwise} \end{cases} \tag{10.4}$$

$$y_\ell^{kvj} = \begin{cases} 1 & \text{If request } k \text{ is served on link } \ell \text{ using the } j^{th} \text{ free window of vehicle } v \\ 0 & \text{Otherwise} \end{cases} \tag{10.5}$$

$$z_k = \begin{cases} 1 & \text{If carsharing request } k \text{ is served} \\ 0 & \text{Otherwise} \end{cases} \tag{10.6}$$

Contrary to the problem in (10.3) where all vehicles had the same link set L, here each vehicle has a different link set in each of its free time windows. Let us keep the set of links for vehicle v during its j^{th} free window in set L_{vj}. Furthermore, we introduce a new set $L_{kvj} = L_k \bigcap L_{vj}$. This set includes all the links that are accessible to both vehicle v during its j^{th} time window, and request k.

The constraint sets that defines this problem are very similar to constraint sets in the previous section, where we routed the autonomous vehicles to satisfy the set of essential trips. Constraint sets (10.7b)-(10.7d) route vehicles within their free time windows. Constraint set (10.7b) ensures that each vehicle at each of its free time windows leaves its origin station after delivering its last passenger. Constraint set (10.7c) ensures that the vehicle reaches its destination location before the departure time of its next scheduled passenger. Constraint set (10.7d) is the flow conservation constraint. Constraint sets (10.7e)-(10.7g) route on-demand requests in the network. These sets of constraints are similar to constraint sets (10.7b)-(10.7d) that route vehicles, with a small variation that not all on-demand requests can be necessarily served. This is reflected in the formulation by replacing 1 on the right hand side of constraint sets (10.3e) and (10.3f) by variable z_k in constraint sets (10.7e) and (10.7g) . Finally, constraint set (10.7h) ensures that each served request is assigned a single vehicle, and each vehicle is assigned to only one request at a time. The objective of the carsharing problem (10.7a) is to maximize the total number of served requests.

$$\text{Minimize} \quad \sum_{k \in M} z_k \tag{10.7a}$$

$$\sum_{\substack{\ell \in L_{vj}: \\ s_i = OS_{v,j}}} x_\ell^{vj} - \sum_{\substack{\ell \in L_{vj}: \\ s_j = OS_{v,j}}} x_\ell^{vj} = 1; \ \forall v \in V, j \in J(v) \tag{10.7b}$$

$$\sum_{\substack{\ell \in L_{vj}: \\ s_j = DS_{v,j}}} x_\ell^{vj} - \sum_{\substack{\ell \in L_{vj}: \\ s_i = DS_{v,j}}} x_\ell^{vj} = 1; \ \forall v \in V, j \in J(v) \tag{10.7c}$$

$$\sum_{\substack{s_i \in S, t_i \in T: \\ \ell = (t_i, s_i, t, s) \in L_{vj}}} x_\ell^{vj} = \sum_{\substack{s_j \in S, t_j \in T: \\ \ell = (t, s, t_j, s_j) \in L_{vj}}} x_\ell^{vj}; \ \forall v \in V, j \in J(v), \forall t \in T, s \in S_e \backslash D_o : \ell = (t_i, s_i, t, s) \in L_{vj} \tag{10.7d}$$

$$\sum_{\substack{v \in V \\ j \in J(v)}} \sum_{\substack{\ell \in L_{kvj}: \\ s_i = OS_k}} y_\ell^{kvj} - \sum_{\substack{v \in V \\ j \in J(v)}} \sum_{\substack{\ell \in L_{kvj}: \\ s_j = OS_k}} y_\ell^{kvj} = 1; \ \forall k \in M \tag{10.7e}$$

$$\sum_{\substack{v \in V \\ j \in J(v)}} \sum_{\substack{\ell \in L_{kvj}: \\ s_j = DS_k}} y_\ell^{kvj} - \sum_{\substack{v \in V \\ j \in J(v)}} \sum_{\substack{\ell \in L_{kvj}: \\ s_i = DS_k}} y_\ell^{kvj} = 1; \ \forall k \in M \tag{10.7f}$$

$$\sum_{\substack{s_i \in S_e, t_i \in T: \\ \ell = (t_i, s_i, t, s) \in L_{kvj}}} y_\ell^{kvj} = \sum_{\substack{s_j \in S_e, t_j \in T: \\ \ell = (t, s, t_j, s_j) \in L_{kvj}}} y_\ell^{kvj}; \quad \begin{array}{l} \forall v \in V, j \in J(v), k \in M, t \in T, \\ s \in S_e \backslash \{OS_k \cup DS_k\} : \ell = (t_i, s_i, t, s) \in L_{kvj} \end{array} \tag{10.7g}$$

$$\sum_{k \in M, \ell \in L_{kvj}} y_\ell^{kvj} \ \leq \ x_\ell^{vj}; \ \forall v \in V, j \in J(v), \ell \in L_{vj} \tag{10.7h}$$

10.3 Solution Method

We formulated the first optimization problem to find the minimum number of autonomous vehicles required to serve a cluster's set of essential trips, and optimally route these vehicles. This problem does not need be solved in real-time, and therefore for problems of moderate size (as we will discuss later) optimization engines such as CPLEX can be used.

The second optimization problem that maximizes the number of served carsharing requests may need to be solved in real-time, as carsharing requests arrive dynamically. In this section, we devise a greedy heuristic algorithm to solve this problem in real-time. The numerical study that follows illustrates the level of efficiency and accuracy of this heuristic algorithm.

The carsharing problem as described in the previous section bears similarities to the family of parallel machine scheduling problems in manufacturing. This class of problems includes a large variety of problems, and is used to find the optimal sequence of using machinery in manufacturing processes. Parallel machine scheduling problems vary in job characteristics (whether there are preemptive or precedence constraints present, fixed/relaxed start or finish time, etc.), machine characteristics (identical or non-identical, serial or parallel, etc.), and the optimality criteria (max number of completed jobs, min makespan, etc.). In the context of our carsharing problem, jobs are carsharing requests, and machinery are the free time windows of drivers. The problem we are trying to solve has the following characteristics:

1. No preemptive or precedence constraints present: Once we fix the schedules of the essential trips, the vehicles' free time windows can be used in any manner, i.e. there is no precedence requirement on the sequence of the carsharing request to be satisfied.
2. Multiple non-homogeneous machines/servers: In our problem each free time window of each vehicle acts as a separate server. Furthermore, our servers are non-homogeneous, meaning that each vehicle at each of its free time windows has distinct origin and

station, as well as start and finish times.

3. Jobs are available during specified time windows, rather than with specific start and finish times: the carsharing requests specify a time window during which a vehicle is required, rather than specify the exact time for start and end of their requests.

4. Set-up cost: In our problem there exist server- and job sequence-dependent set-up costs. Because vehicles have to travel to location where they are requested, which vehicle is to be assigned to a request, and the sequence of requests assigned to a vehicle, all play a role.

5. Objective: maximizing the number of served jobs (satisfied carsharing requests).

There is an extensive amount of literature on machine scheduling (Hall and Sriskandarajah 1996; Cheng et al. 2004). Rabadi et al. (2006) propose heuristics to solve the non-preemptive unrelated parallel machine scheduling problem, in which machine- and job sequence-dependent setup times are considered, but jobs are all assumed to be available at time zero. Gabrel (1995) proposes heuristics to solve the problem of scheduling non-preemptive jobs with an interval for starting time, on identical parallel machines. To the best of our knowledge, there is no study that combines both characteristics (set-up costs, and time windows for jobs), that can be used to solve the carsharing problem formulated in the previous section.

10.3.1 Heuristic Algorithm to Solve the On-Demand Carsharing Problem

The heuristic algorithm described in this section is based on the earliest finishing time (EFT) heuristic originally designed to solve the interval scheduling problem. In the interval scheduling problem, there is a machine that needs to complete the maximum number of jobs possible. Each job has a specific start and finish time. At each step, the EFT heuristic selects the job with the earliest finishing time that does not conflict with the previously selected jobs. The EFT heuristic yields optimal solutions.

The carsharing problem we need to solve is substantially more complicated than the interval scheduling problem. In fact, it is easy to see that the carsharing problem is NP-Hard. Here, we modify the EFT heuristic, and tailor it to solve the carsharing problem. Our proposed algorithm is displayed in Algorithm 3.

In the mathematical program in (10.7), we used the tuple (v, j) to refer to the j^{th} free time window of vehicle v. In the interest of simplifying notation, we treat each free time window of each vehicle as a separate vehicle $v' \in V'$, where $V' = (v, j)|v \in V, j \in J(v)$.

Algorithm 3 On-demand vehicle allocation

Allocates carsharing requests using the idle autonomous vehicles

```
1. Initialize: $\forall v' \in V'$
   $Loc(v') = OS_{v'}$
   $Time(v') = ED_{v'}$
2. Find the set of feasible requests $R(v')$, $\forall v' \in V'$
   $\forall k \in R:$
       If   $Max\{Time(v') + shp^1(Loc(v'), OS_k), ED_k\} + shp(OS_k, DS_k) \leq LA_k$
               $R(v') = R(v') \cup \{k\}$
3. Find the matched request and driver by studying the minimum finishing time for all combinations of
   vehicles and requests $\big(\forall v' \in V', k \in R(v')\big)$
   $(v'^*, k^*) = Argmin_{v' \in V', k \in R(v')} \big\{Max\{Time(v') + shp(Loc(v'), OS_k), ED_k\} + shp(OS_k, DS_k)\big\}$
4. Update sets
   $Loc(v'^*) = DS_{k^*}$
   $Time(v'^*) = Max\{Time(v'^*) + shp(Loc(v'^*), OS_{k^*}), ED_{k^*}\} + shp(OS_{k^*}, DS_{k^*})$
5. Delete $k^*$ from $R(v'), \forall v' \in V'$.
6. Update $R(v'^*)$ based on rider travel time windows:
   $\forall k \in R(v'^*):$
       If   $Max\{Time(v'^*) + shp(Loc(v'^*), OS_k), ED_k\} + shp(OS_k, DS_k) > LA_k\big\}$
         $R(v'^*) = R(v'^*) \backslash k$
   Go to step 3.
7. Stopping Criteria: $\forall v' \in V', R(v') = \emptyset$.
```

[1] $shp(i, j)$: shortest path travel time between i and j

In the first step of the algorithm, we initialize two sets of arrays. The first array, $Loc(v')$,

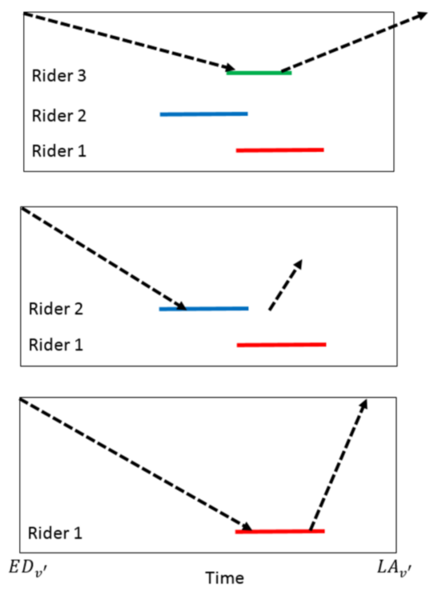


Figure 10.3: Determining the set of feasible requests $R(v')$ for vehicle v'

indicates the current location of vehicle v'. The second array, $Time(v')$, indicates the time vehicle v' becomes idle (available). We initialize the location array Loc for each vehicle $v' \in V'$ with the origin station of the vehicle, and the time array $Time$ with the earliest departure time of the vehicle.

The algorithm starts by determining the set of feasible carsharing requests for each vehicle. In order for a request to be feasible for a vehicle, the vehicle should be able to drive from its current location to the request's origin station, and get there at or after the start of the request's time window, stay in possession of the requester for the requested duration of time, and finally arrive at its own destination (the pick-up location of its next scheduled essential trip) before its latest arrival time. Figure 3(b) studies the feasibility of three carsharing requests for a vehicle. The boundaries of the boxes show the free time window of the vehicle, and the line (blue, red, green) associated with each request marks its time window. The first request (at the bottom) is feasible for the vehicle. The vehicle arrives at the request's origin location after the request's earliest departure time, is able to stay in possession of the requester for the demanded duration that ends before the request's latest arrival time, and travels to its destination station within its time window. The second and third requests, however, are not feasible for the vehicle. In the case of the second request, the vehicle cannot stay in possession of the requester for the duration of the request, and in the case of the third request, the vehicle cannot go back to its own destination station after finishing serving the request. In the third step of the algorithm, we find finishing times for all combinations of vehicles and their set of feasible requests. The finishing time of vehicle vˆ' serving request k includes the time required for the vehicle to arrive at the request's origin location, and then stay in possession of the requester for the demanded period of time. Note that if the vehicle arrives at a requested location before the start of the request's time window, it has to wait until the start of the time window. The vehicle and request pair that lead to the earliest finishing time will be selected and matched together. In step 4, the location of the matched vehicle will be updated to the location of the destination of the matched request, and the time array of the assigned vehicle will be updated to the drop-off time of the rented out vehicle. In step 5, the matched request in step 3 will be eliminated from the set of available requests to all vehicles. Furthermore, since the time window and location of the matched vehicle in step 3 have been updated, the set of feasible requests for this vehicle needs to be updated as well. The algorithm stops when all vehicles have empty sets of feasible requests.

10.4 Real-World Implementation

We implemented the SVOU program for a sample of households in San Diego County, using data from the 2000-2001 California statewide household travel survey (Casas, 2002). In this survey, Caltrans collected travel data from 17049 volunteer households in California.

After cleaning the data by eliminating records with incomplete or contradicting information, a total of 1184 households residing in the city of San Diego were remained. For these households, detailed information on the number of household members, the number of vehicles owned by households, logged trips of each member during a working day along with the purpose of each trip were available, among other information.

After cleaning the trip sets, we determined the set of essential trips for each household based on the information on the purpose of trips. We categorized trips concerning work, school, childcare, medical, fitness, community meetings, volunteer activities, visiting friends and family, and entertainment activities as essential (core), and the rest of the trips as non-essential. Among the 1184 households, 573 of them did not report any essential trips during the survey day, and therefore were not considered for the shared use and ownership program. These households, however, were taken into consideration for the car rental service to serve their non-essential trips. The 1184 households made a total of 3306 trips, 1624 (49%) of which were essential trips.

The first step in implementing the program is to cluster households. Each cluster should include a number of households with enough commonalities that would interest them to participate in the shared vehicle ownership and use program together. Various parameters can be used to determine a suitable cluster for a household, including home location, demographics and social status of household members, level of spatiotemporal proximity of trips between households, and income level, to name a few. In this study, we cluster households at different levels and using different criteria. In sections 10.4.1 and 10.4.2 we study the impact of clustering households based on the proximity of their home locations and the degree of overlap between their trip sets, respectively. In section 10.4.3, we study two extreme clustering approaches and estimate upper and lower bounds on the potential savings on the number of vehicles and VMT in our sample.

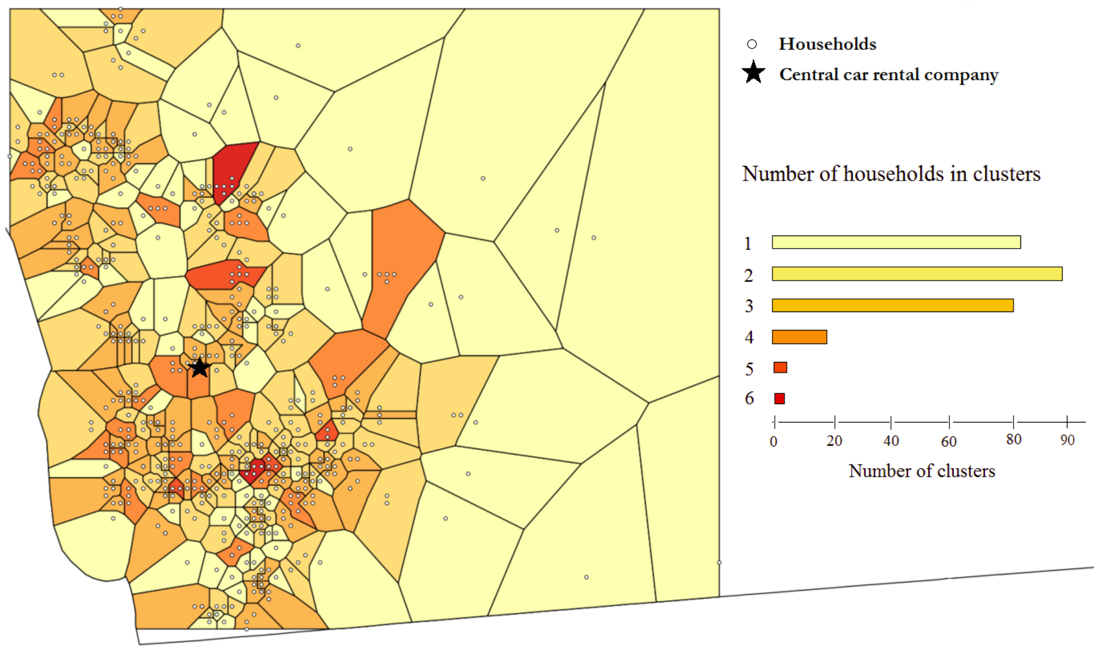


Figure 10.4: Clusters of households. Households in each cluster are assumed to share ownership of a set of autonomous vehicles

10.4.1 Location-Based Clustering

In our first implementation, we use agglomerative clustering, an unsupervised learning method, to group households based on their home location (Steinbach et al., 2000).

Figure 10.4 displays the resulting 277 clusters of households. These clusters are distinguished by color based on the number of their household members. Figure 10.4 also displays the distribution of number of households in clusters. About 80% of clusters have three or fewer household members, which makes managing of shared vehicles an easier task. About 30% of the households are geographically isolated from others, and therefore remain as singleton clusters. Figure 10.4 also illustrates the Voronoi polygons attributed to clusters. These polygons suggest to which cluster a prospective household looking to join the program would belong based on its home location.

For each cluster, we solve the optimization problem (10.3) to find the optimal number of vehicles required to cover the cluster's set of essential trips. All problems are solved on a PC

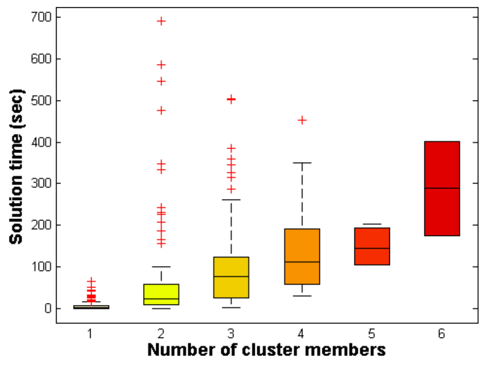


Figure 10.5: Solution time (sec) of finding the optimal number of vehicles and vehicle itineraries for each cluster size

with Core i7 3GHz and 8GB of RAM, using AMPL modeling language and CPLEX 12.6.00 solver with standard tuning. Solution times are displayed in Figure 10.5. Not surprisingly, the solution times increase with cluster size; However, they remain within a reasonable range for a problem that does not need to be solved in real-time.

The solution suggests that a total of 379 vehicles are required to serve all the essential trips by all clusters (including the one-household clusters). Note that households are still in need of transportation for their non-essential trips. Therefore, this number serves as a lower bound to the total number of required vehicles.

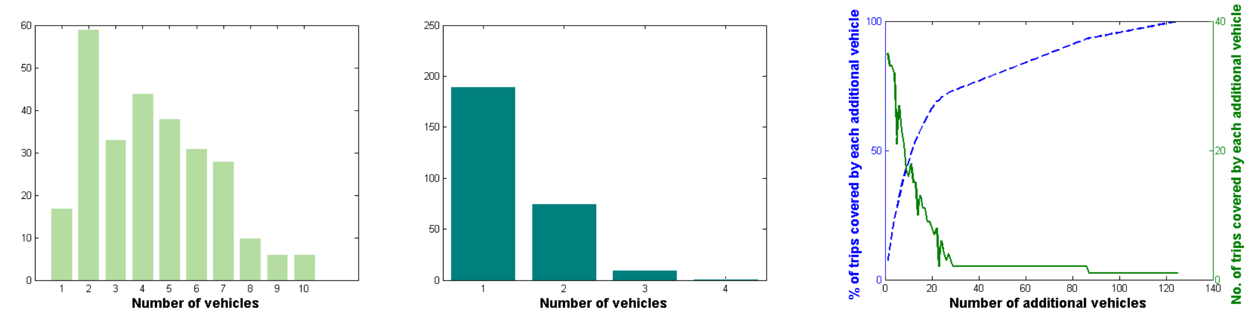
Figure 10.6a shows the distribution of number of vehicles households owned in 2000. This figure suggests that the majority of households owned at least 2 vehicles. Figure 10.6b shows the distribution of minimum number of vehicles needed in the clusters, obtained from our methodology. This figure shows that about 65% of clusters need no more than one vehicle to serve their essential trips. No cluster needs more than 4 vehicles.

After forming clusters of households and routing the set autonomous vehicles owned by each cluster to serve the cluster's set of essential trips, we need to address the non-essential trips. One possibility is to use a central car rental system that uses the idle autonomous vehicles owned by clusters to serve non-essential trips of the entire population. The question is, what percentage of the non-essential trips can be served by such a centralized system, and how many additional vehicles need to be owned by the rental service provider to serve the entire set of non-essential trips. Note that not only the 611 households who participate in the SVOU program, but also the 573 households which did not report any essential trips need to have their non-essential trips served.

Using Algorithm 3 to rent out the 379 autonomous vehicles owned by clusters with the goal of serving as many non-essential trips as possible, we manage to serve 63 of the non-essential trips. We then use a variation of Algorithm 3 to find the additional number of vehicles the rental company needs to own in order to serve the remaining 37% of the non-essential trips. We assume a yard for the rental service provider located strategically in the network (marked with a star symbol in Figure 10.4) where there is a very high density of trip origin and destination locations. We increment the number of vehicles one at a time, having each vehicle start its trip from the yard, to which it returns at the end of the day. We assign trips to each newly added vehicle using Algorithm 3 until all trips are served.

The results suggest that a total of 125 vehicles need to be owned and managed by the rental service provider to serve the rest of the non-essential trips. All these vehicles, however, do not have the same contribution in terms of the number of served trips. Figure 10.6c displays the incremental percentage and number of trips served by each additional vehicle. This figure suggests that using only 30 vehicles, the rental service provider can serve 75 of the remaining non-essential trips. The remaining 95 vehicles each serve only 1 or 2 trips with origins and/or destinations in remote areas. In fact, using cluster-owned vehicles in their idle times, and owning 30 vehicles, the rental company can serve more than 90 of its demand of non-essential trips.

Comparing the count of 504 vehicles that could serve the entire transportation demand against the 2194 vehicles owned by households suggests that the shared ownership and use program has the potential to have significant impacts on vehicle ownership.



(a) Frequency distribution of the number of vehicles owned by cluster members in year 2000

(b) Frequency distribution of the number of autonomous vehicles required for clusters under the shared vehicle ownership and use program

(c) Number of additional vehicles required to serve non-essential trips

Figure 10.6: Impact of the shared vehicle ownership and use program on vehicle ownership

The savings in the number of vehicles in the proposed system originate from three different sources: (1) introduction of autonomous vehicles, (2) shared ownership of these vehicles, and (3) ridesharing within clusters. It would be interesting to observe how much of the savings can be attributed to each source. Toward this end, we consider two additional cases. In the first case, we study the impact of households trading their current vehicles for the optimal number of autonomous vehicles. In the second case, we allow households to form clusters in order to share the ownership of vehicles. The third and most complete case considers the shared ownership and shared use of vehicles among the members of each cluster, as suggested by the SVOU program.

Table 10.1 summarizes the number of required vehicles, and the total and average vehicle miles traveled (VMT) for each case. Note that for now we assume that there is parking space available to vehicles when and where required. In case 1, where ordinary vehicles are replaced with autonomous vehicles, a total of 787 vehicles need to be purchased by the households in our sample. Not surprisingly, total and average VMT increase, since the number of autonomous vehicles is substantially less than the number of ordinary vehicles. In the second case, allowing shared ownership of autonomous vehicles leads to a 30% decrease in vehicle ownership compared to case 1, although this decrease in the number of vehicles

Table 10.1: Impact of different elements of the shared ownership and use program on vehicle ownership and vehicles miles traveled using distance-based clustering

	Base Case: Year 2000 data	Case 1: Autonomous vehicles	Case 2: Autonomous vehicles + Shared ownership	Case 3: Autonomous vehicles + Shared ownership + Shared use
No. of vehicles	2,194	787	528	504
Total VMT	29,959	62,980	64,836	65,610
Average VMT	13.6	80	122	130

comes at the cost of a 50% increase in the average VMT for each vehicle. The increase in total VMT, however, is not substantial. Finally, in the third case where shared ownership and use of autonomous vehicles is studied, an additional 5% decrease in vehicle ownership compared to case 2 can be witnessed, accompanied by a slight increase in both total and average VMT.

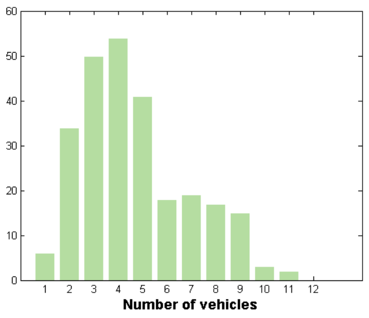
10.4.2 Clustering Based on Trip Overlap

An alternative way of clustering households in based on the overlap between household trips. For each household pair, we compute the *degree of compatibility* between their trips. For a given pair of households h_1 and h_2, we define an $n_1 \times n_2$ matrix, where n_1 and n_2 denote the number of essential trips by the two households, respectively. Each cell c_{ij} in this matrix takes the value of 1 if trips i and j can be fulfilled using the same vehicle, i.e., either one of the two following conditions holds: (1) the vehicle can fulfill the trips sequentially (fulfill the first trips, drive to pick up the passenger for the second trip and fulfill the second trip), or (2) the driver can fulfill the trips concurrently (perform the pick-up task for the both trips, followed by performing the drop-off task for the two trips). If neither of these two conditions is satisfied, the cell c_{ij} will be assigned the value of 0, implying that the two trips are incompatible. The summation of all elements in matrix c is what we define as the degree of compatibility between households $h_1$and h_2. We use agglomerative clustering based on the degree of compatibility between households to group households into 261 clusters.

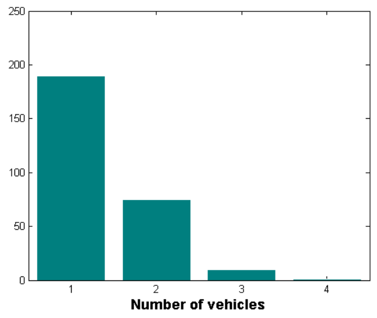
A total of 340 autonomous vehicles are required to serve the essential trips of the 261 clusters. Figures 10.7a and 10.7b compare the distribution of vehicles owned by households before and after implementing the shared ownership and use program. It is interesting to note that more than half of the clusters need only one autonomous vehicle, whereas in year 2000 data the majority of clusters own four or more vehicles. Figures 10.7c and 10.7d display the distribution of number of households in clusters, and the solution times to optimize the itineraries in clusters, respectively. Similar to the distance-based clustering, the solution times remain within a reasonable range.

During their idle times, essential vehicles managed by the rental service provider can serve 57% of the non-essential trips of the entire population. The company needs to own 128 additional vehicles to serve the remaining 43% of the non-essential trips. Figure 10.7e shows the contribution of each additional vehicle to serving the remainder of non-essential trips. This figure suggests that although 128 vehicles are required to cover the entire set of non-essential trips, the first 27 vehicles can serve more than 70% of the remaining set of non-essential trips, and around 90% of the entire set of non-essential trips.

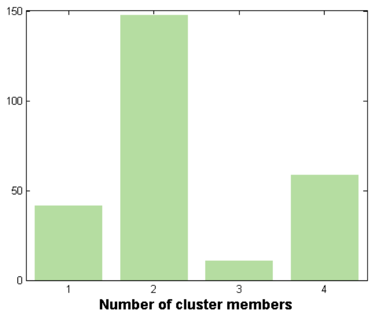
Similar to the previous section, we study the impact of each component of the shared ownership and use program on vehicle ownership and VMT. Results are displayed in table 10.2. The same decreasing trend in vehicle ownership that was observed in distance-based clustering can be witnessed in trip-overlap clustering as we move from case 1 to case 3. For each case, however, the total number of vehicles in trip-overlap clustering is less than that of distance-based clustering. This is an intuitive finding, since in trip-overlap clustering we are grouping households whose trips have higher degrees of compatibility, and therefore can be served using fewer vehicles. Fewer vehicles translates into higher total and average VMT. One interesting observation is that in case 3 where we allow households in a cluster to rideshare, the total and average VMT are less than case 2 in which ridesharing is not allowed. In distance-based clustering, adding the ridesharing capability slightly increased



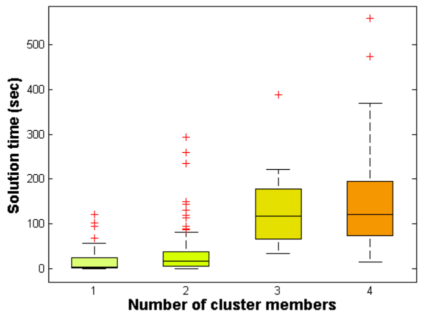
(a) Histogram of number of vehicles currently owned by cluster members



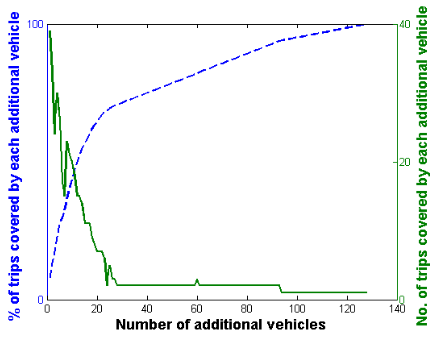
(b) Histogram of optimized number of autonomous vehicles required for clusters



(c) Histogram of optimized number of autonomous vehicles required for clusters



(d) Solution time (sec) of finding the optimal number of vehicles and vehicle itineraries for each cluster size



(e) Number of additional vehicles required to serve the entire set of non-essential trips

Figure 10.7: Impact of the shared ownership and use program on vehicle ownership

Table 10.2: Impact of different elements of the shared ownership and use program on vehicle ownership and vehicles miles traveled using trip overlaps for clustering

	Base Case:	Case 1:	Case 2:	Case 3:
	Year 2000 data	Autonomous vehicles	Autonomous vehicles + Shared ownership	Autonomous vehicles + Shared ownership + Shared use
No. of vehicles	2,194	787	497	468
Total VMT	29,959	62,980	102,790	73,841
Average VMT	13.6	80	207	158

the total and average VMT.

10.4.3 Level of Clustering

In sections 10.4.1 and 10.4.2 we studied the impact of clustering households based on two different criteria. In this section, we present two additional and more extreme clustering approaches to provide some bounds on the impact of shared vehicle ownership and use program for our sample of households. Results are displayed in table 10.3.

The first extreme approach is to conduct no clustering at all, and assume each households as a singleton cluster. Our sample includes a total of 611 households with non-empty sets of essential trips. Our analysis shows that these households need 678 vehicles to cover their essential trips. To satisfy the non-essential trips of all 1184 households in our data set, The car rental service provider needs to own 109 separate vehicles in addition to having access to shared vehicles during their idle times.

By simply replacing ordinary vehicles by the optimal number of autonomous vehicles, the total number of vehicles owned by the entire sample reduces from 2194 to 787. This 2.8 fold reduction in the number of vehicles, however, comes with higher VMT for vehicles, as discussed in the previous section. The amount of increase in VMT, however, depends highly on the availability of parking. Here, we discuss two extreme scenarios on parking availability. In the first scenario, parking is available when and where required, and in the second scenario, there is no parking available, i.e. vehicles have to drive around until it is time for their next pick-up job. In the latter scenario, the unavailability of parking is not limited to physical availability, but the affordability of the available parking spaces as well. If parking is priced highly, it might be a financially wiser alternative for vehicles to drive around.

Our analysis shows that the availability and affordability of parking space have substantial impact on the total VMT. For the households in our sample, we find a 5-fold increase in the total and average VMT when there is not access to affordable parking. This result, however, is obtained under the assumption that introduction of autonomous vehicles does not change households' travel patterns.

In the second extreme case, we group all households into a single cluster. In this case, a total of 258 vehicles would suffice to serve all trip requests, which is a 3-fold reduction in the number of vehicles compared to the first extreme case, where each household formed a separate cluster. An interesting observation is that even under the assumption of unavailability of affordable parking, this case leads to substantial savings in the total VMT. This observation has two implications. First, as the number of households participating in the program and hence the average cluster size increases, the impact of parking availability becomes less prominent. Second, the substantial reduction in VMT along with a decrease in the number of vehicles observed under the single-cluster scenario demonstrate the significant benefits that the marriage of ridesharing services and autonomous vehicles can provide in transportation systems.

Our two realistic clustering approaches from sections 10.4.1 and 10.4.2 lie between the two extreme cases studied above. Although the difference between the distance-based and trip overlap-based clustering is not substantial, we expect the benefits of clustering households based on trip overlap to become more prominent with higher participation rates.

Table 10.3: Impact of level of clustering and parking availability of vehicle ownership and VMT

Clustering method	Household-based (no clustering)	Geographical distance	Trip overlap	Single cluster
No. of clusters	611	277	261	1
Avg. cluster size	1	2.21	2.34	1184
Min no. of autonomous vehicles owned by clusters	678	379	340	258
Percentage of non-essential trips covered	63%	63%	57%	100%
Additional no. of vehicles required	109	125	128	0
Total number of vehicles	787	504	468	258
Parking not available:				
Optimized total VMT	335,640	305,800	286,060	158,020
Current avg. VMT	13.6	13.6	13.6	13.6
Optimized avg. VMT/shared vehicle	450	715	734	612
Optimized avg. VMT/additional vehicle	278	278	286	0
Optimized avg. VMT/vehicle	426	607	611	612
Parking available when and where desired:				
Optimized total VMT	62,980	65,610	73,841	76,134
Optimized avg. VMT/shared vehicle	73	134	174	295
Optimized avg. VMT/additional vehicle	121	119	115	0
Optimized avg. VMT/vehicle	80	130	158	295

Among the clustering approaches discussed above, the last scenario which considers a single cluster renders the most benefits. In this scenario, a regional shared-mobility service provider owns and manages a fleet of autonomous vehicles that serves the entire transportation needs of a region. We can, in fact, expect this type of shared-mobility service to emerge before autonomous vehicles are offered for private ownership, for two reasons. First, in order for autonomous vehicles to be ready for consumer use, adequately-detail maps of the entire United States road network need to be collected. The type of maps required for this purpose need to include far more details than currently contained in street maps maintained by Google and the like. To have fleets of autonomous vehicles operate in certain urban regions would require far less efforts to collect street data on the targeted areas. Second, a fleet of autonomous vehicles used in a shared-mobility context in populous areas would have very high utilization rate, which amounts to very low cost to the consumer. High demand makes such shared-mobility services less sensitive to the high cost of autonomous technology, and the inexpensive transportation alternative provided to the consumers may render vehicle ownership in certain urban regions obsolete.

This type of regional shared-mobility service, however, might not be accessible or attractive to everyone. It might not be a financially feasible strategy for shared-mobility service providers to target households who reside in remote areas. In addition, there will always be individuals who would like to ensure their privacy, flexibility, or timeliness by owning private vehicles. Such individuals would seek other clustering approaches, or at extreme case, form individual clusters.

10.5 Discussion

Throughout this chapter, we made the assumption that the introduction of autonomous vehicles does not impact household travel behavior. However, autonomous vehicles can change household travel patterns in multiple ways. Having access to a new technology that allows individuals to make use of their time while traveling can encourage longer trips that would have otherwise been avoided due to the burden of driving. Moreover, access to autonomous vehicles could induce higher number of trips. With self-driving vehicles, trip chaining (which is currently a necessity to many households) would not be as essential. Self-driving cars can transport household members without a valid driver's license and perform activities such as parking or refueling, making current trip chains smaller, changing travel patterns, and increasing the number of trips. Longer and more frequent trips impose higher costs on the transportation infrastructure, and can cancel out some of the benefits that autonomous cars introduce by reducing the number of vehicles.

The results of our study implies that availability of affordable parking is a major determinant of the total VMT and the consequent cost to the transportation infrastructure and the environment. We showed, however, that this impact depends on the degree and type of clustering. With the right type of clustering and larger cluster sizes, total VMT stays within a reasonable range. In the best case scenario where a central shared-mobility service provider serves the entire transportation demand of a region, the change in the VMT becomes close to non-existent.

In spite of a possible increase in VMT, autonomous vehicles will significantly reduce the adverse impact of transportation on the environment by targeting two main sources of emission dissemination. Autonomous vehicles can reduce the number of vehicles required by orders of magnitude, as demonstrated in this study. In addition, these vehicles significantly increase network capacity, as we will discuss next. The combination of these two factors can reduce, and in some regions completely eliminate, the stop-and-go conditions that are a major contributor to vehicle emissions.

In addition to fewer vehicles, autonomous vehicles can further contribute to congestion-relief due to their ability to communicate. Tientrakool (2011) shows that a highway populated with a mix of autonomous and ordinary vehicles can experience substantial increase in capacity depending on the percentage of communicating vehicles in the traffic mix. Her study suggests that when less than 30% of the traffic mix can communicate, the resulting rate of change of capacity is rather slow. With penetration rate of $30\% - 85\%$ the rate of change of improvement in capacity increases, and when the percentage of communicating vehicles in the traffic mix exceeds 85%, the resulting rate of change of increase in capacity improves very quickly, to the extent that at the 100% penetration rate, the capacity reaches more than $10,000$ vehicles/hours/lane. The increase in capacity is caused by lower inter-vehicle gaps that need to be maintained. These findings further highlight the positive impacts of having a centralized shared-mobility service provider in populous urban areas.

Autonomous vehicles eliminate the possibility of human error, which is the leading cause for the majority of traffic collision fatalities. An autonomous driver will never get distracted, fall asleep, or drive under the influence. Furthermore, autonomous vehicles can make split-second decisions based on probabilistic models fed by far more complete information than a human driver can have access to, while a human driver needs to take longer to make a decision based on in-complete information. As the percentage of autonomous vehicles in the traffic mix increases, so does the level of safety as the network becomes more deterministic. This is another benefit that can be experienced with implementation of a centralized shared-mobility system.

To conclude, although change in travel patterns and increase in VMT under certain conditions may lead to higher costs to the transportation system, the many benefits of autonomous vehicles described in this chapter more than outweigh the possible downfalls. Furthermore, higher contribution rates in the SVOU program would exponentially increase the benefits. Note that the sample we used for this study contained only 0.1% of the population, which is far less than the participation rate expected for such services. Despite the small sample size, we showed that in the case of a central shared-mobility service provider we can achieve 10-fold reductions in the number of vehicles. These results push the previous bounds suggested by autonomous taxi studies that have estimated a 6-10 fold reduction in the number of vehicles. Much of the the higher success rate of the SVOU program proposes in this chapter can be attributed to its “shared use” component.

Finally, introduction of autonomous vehicles in the traffic mix may call for some policy adjustments to reduce the possible adverse impacts of this new technology. Our study points out that parking availability and cost play important roles on the potential environmental and congestion-relief benefits expected from autonomous vehicles. A grid-lock for an autonomous vehicle that is traveling with no purpose is equivalent to free parking. The temptation to drive around when idle becomes more strong with alternative-fuel autonomous vehicles. Under current regulations, roads and highways are funded by gas tax, and therefore the only driving related cost alternative-fuel autonomous vehicles would have to bear is the depression cost associated with higher VMT. To avoid such behavior, changing from gas-based to VMT-based tax might be a necessary policy adjustment.

Chapter 11

Conclusion

This dissertation introduces a mathematical framework for modeling the problem of matching riders and drivers in a peer-to-peer (P2P) ridesharing system under uncertainty for trip requests. The focus is on the most general form of ride-matching, namely many-to-many matching in which each driver can carry multiple riders at each point in time, and each rider can transfer between multiple vehicles, although more efficient algorithms for more restrictive forms of ride-matching are also provided. The contributions of this dissertation are primarily methodological. However, multiple numerical experiments and case studies provide insights on successful implementation strategies of ridesharing systems that can be useful to decision makers.

To mathematically model the problem, we discretize the network into a set of stations. These are pre-specified locations where individuals can start and end their trips, and riders can transfer between vehicles. In addition, we discretize the study time horizon into small time periods and formulate the problem of P2P ride-matching on a time-expanded network, where a node is defined as a tuple of time period and station. The ride-matching problem on a time-expanded network is in fact an integer (binary) flow optimization problem. The basic formulation is quite efficient and powerful, and its variants can be used for a wide variety of problems of the future.

To model a P2P ridesharing system, we adopt a rolling time-horizon approach, where the system is re-optimized periodically at pre-specified re-optimization times. This modeling approach allows us to increase the performance and reliability of the system by incorporating the data on the most recent trip requests and travel times in the decision making process.

At each re-optimization time, the system solves a stochastic ride-matching problem. This problem includes stochastic requests that are not registered, but predicted to arrive during the first re-optimization period, and deterministic trips registered in advance to occur during the next few re-optimization periods. The solution to this problem provides itineraries for the deterministic riders and drivers, as well as contingency plans on drivers' routes in case the predicted trip requests turn into registered trips.

A decomposition algorithm is proposed to solve the deterministic many-to-many matching problem, by means of iteratively solving smaller sub-problems. This decomposition algorithm can be used to efficiently generate a set of scenarios required to solve the deterministic re-formulation of the stochastic problem.

The deterministic re-formulation of the stochastic program can be solved using an L-shaped decomposition, where sub-problems are always feasible. The feasibility of sub-problems makes it possible to continuously improve the solution through iterations, and have a high-quality solution in case the algorithm needs to be stopped prematurely due to the real-time nature of the problem. Furthermore, due to the fact that sub-problems in each iteration of the L-shaped decomposition are independent of each other, it is possible to speed up the solution process by implementing parallel computations, and overcome the burden of solving a large-scale problem by distributing the computations among the users' mobile devices.

Although solving a stochastic problem allows the system to take into consideration trip requests that may arrive at the system at some later point in time, there could still be real-time requests that cannot be successfully predicted. Such real-time requests can be served using the many-to-one dynamic programming algorithm proposed in this dissertation.

Transportation systems in general serve requests on a first-come first-served basis. Whether it is due to lack of computational resources, the real-time nature of requests, or the quest to be "fair", this pre-specified order of serving requests does not allow the system to realize its full potential in terms of the number of served requests. In this dissertation, we propose what we call "P2P ride exchange", a bi-lateral trade mechanism that allows previously matched riders to sell their itineraries to un-matched riders, in exchange for a less attractive itinerary, and a monetary compensation. Our numerical experiments suggest that this mechanism can improve the matching rate by up to 15%.

The large variety of numerical experiments simulating the deterministic and stochastic ridesharing systems that are presented in this dissertation have provided interesting insights on the behavior of such systems under different circumstances, and the insights therein can potentially lead to efficient design considerations in future practical implementations.

No matter the matching method, if riders and drivers are assumed to form two mutually exclusive sets, the matching rates are maximized when the number of riders and drivers are about the same. However, under a more realistic scenario where individuals initially join the system as riders and switch to taking a driver's role only if they cannot be successfully matched as riders, the highest matching rates are obtained under the highest ratio of riders. This observation has strong policy implications, suggesting that providing incentives for individuals to participate in a ridesharing system initially as riders gives the system a higher chance of reaching a critical matching rate required for sustainable operations.

The multi-hop property of the proposed algorithms allows for them to be used in multi-modal settings. This capability was put into play in a case study in Los Angeles County, where

P2P ridesharing was used to feed the LA metro red line, under the assumption of real-time trip request arrivals. This study demonstrates the scalability of the dynamic programming algorithm proposed in this dissertation, and provides insights on the performance of the system under different levels of supply. Although it might seem counter-intuitive in the first glance, for a fixed number of trip requests, there is an optimal range for the number of drivers that leads to the highest performance levels. Going beyond this range does not improve the matching rate, but deteriorates sustainability-related indices such as vehicles miles traveled.

The Los Angeles case study also provides insights on the pricing of ridesharing systems, and demonstrates how appropriate pricing levels could vary from region to region depending on the residents' financial welfare and their values of time. Furthermore, this study investigates under what levels of fares a ridesharing system can be budget-balanced, and operate without need for subsidies.

Finally, this dissertation explores the impact of shared autonomous mobility on private vehicle ownership and the transportation infrastructure, and makes recommendations on the policy and land-use adjustments that should be rigorously studied to provide the foundation for seamless transition to autonomous technology.

Studying the impact of implementing a shared ownership and ridership program in the San Diego county suggests that self-driving vehicles can introduce a much more flexible and inexpensive form of shared-mobility in certain populous regions, rendering vehicle ownership and public transit in its current form in such regions obsolete. For other areas with less dense populations, replacing ordinary vehicles with autonomous cars, especially under a shared ownership program, can still introduce benefits in terms of vehicle ownership. Our study also suggests that the extent of environmental and congestion-relief benefits expected from autonomous vehicles depend on some operational and deployment strategies, such as availability of affordable parking.

In conclusion, the research developed a framework for rideshare systems that is sound in its theoretical foundations and formulations, as well as efficient in the algorithmic implementations, and demonstrated through numerical studies the possibilities for its use. It is hoped that this framework will be of use in future for transforming the current state of urban travel towards a better world with much fewer empty seats traveling in vehicles, the goal behind this research effort.

Bibliography

Abou-Zeid, M., M. Ben-Akiva, M. Bierlaire, C. Choudhury, and S. Hess (2010). Attitudes and value of time heterogeneity. *Applied Transport Economics-A Management and Policy Perspective. De Boeck Publishing*, 523–545.

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang (2009). Sustainable passenger transportation: Dynamic ride-sharing. *ERIM Report Series Reference No. ERS-2010-010-LIS*.

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research 223*(2), 295–303.

Agatz, N., A. L. Erera, M. W. Savelsbergh, and X. Wang (2011). Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological 45*(9), 1450 – 1464.

Baldacci, R., M. Vittorio, and M. Aristide (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research 52*(3), pp. 422–439.

Böckmann, M. (2013). The shared economy: It is time to start caring about sharing; value creating factors in the shared economy. *University of Twente, Faculty of Management and Governance*.

Braekers, K., A. Caris, and G. K. Janssens (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological 67*(0), 166 – 186.

Business Insider (2015). Uber Just Released Its First Report On Its Drivers Here Are The Numbers. Internet: http://www.businessinsider.com/uber-driver-data-report-2015-1. Accessed: 2015-06-20.

Carnes, T. A., S. G. Henderson, D. B. Shmoys, M. Ahghari, and R. D. MacDonald (2013). Mathematical programming guides air-ambulance routing at Ornge. *Interfaces 43*(3), 232–239.

Carson, B. (2016). Report: Uber was on track to top $ 1.5 billion in revenue last year. `http://www.businessinsider.com/report-uber-15-billion-revenue-in-2015-2016-1`. Accessed 6/29/2016.

Casas, J. (2002). 2001 california statewide household travel survey final report. *California Department of Transportation.*

Cheng, T. E., Q. Ding, and B. M. Lin (2004). A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research 152*(1), 1–13.

Coltin, B. and M. Veloso (2014). Ridesharing with passenger transfers. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 3278–3283. IEEE.

Cordeau, J.-F. and G. Laporte (2007a). The dial-a-ride problem: models and algorithms. *Annals of Operations Research 153*(1), 29–46.

Cordeau, J.-F. and G. Laporte (2007b). The dial-a-ride problem: models and algorithms. *Annals of Operations Research 153*(1), 29–46.

Cortés, C. and R. Jayakrishnan (2002). Design and operational concepts of high-coverage point-to-point transit system. *Transportation Research Record: Journal of the Transportation Research Board* (1783), 178–187.

Cortés, C. E., M. Matamala, and C. Contardo (2010). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research 200*(3), 711–724.

Curvy Road (2000). http://www.curvyroad.com. Accessed July 22, 2015.

Dailey, D., D. Loseff, and D. Meyers (1999). Seattle smart traveler: dynamic ridematching on the world wide web. *Transportation Research Part C: Emerging Technologies 7*(1), 17–32.

Di Febbraro, A., E. Gattorna, and N. Sacco (2013). Optimizing dynamic ride-sharing systems. *Presented in the TRB 2013 annual meeting.*

Dusan, T. and D. Mauro (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. In *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*, pp. 51–60.

Fagnant, D. J. and K. M. Kockelman (2014). The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies 40*, 1–13.

Furuhata, M., K. Daniel, S. Koenig, F. Ordonez, M. Dessouky, M.-E. Brunet, L. Cohen, and X. Wang (2015). Online cost-sharing mechanism design for demand-responsive transport systems. *Intelligent Transportation Systems, IEEE Transactions on 16*(2), 692–707.

Furuhata, M., M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological 57*, 28–46.

Gabrel, V. (1995). Scheduling jobs within time windows on identical parallel machines: New model and algorithms. *European Journal of Operational Research 83*(2), 320–329.

Ghoseiri, K. (2013). Dynamic rideshare optimized matching problem. *Ph.D. Dissertation at University of Maryland*.

Giuliano, G., R. Hall, and J. Golob (1995). Los angeles smart traveler field operational test evaluation.

Hagerty, K. M. and W. P. Rogerson (1987). Robust trading mechanisms. *Journal of Economic Theory 42*(1), 94–107.

Hall, N. G. and C. Sriskandarajah (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research 44*(3), 510–525.

Haselkorn, M., J. Spyridakis, C. Blumenthal, S. Michalak, B. Goble, and M. Garner (1995). Bellevue smart traveler: Design, demonstration, and assessment. final technical report. Technical report.

Heinrich, S. (2010). Implementing real-time ridesharing in the san francisco bay area. *Ph.D. Dissertation at San Jose State University*.

Herbawi, W. and M. Weber (2011a). Ant colony vs. genetic multiobjective route planning in dynamic multi-hop ridesharing. *in Tools with Artificial Intelligence (ICTAI), 2011 IEEE Conference*.

Herbawi, W. and M. Weber (2011b). Comparison of multiobjective evolutionary algorithm for solving the multiobjective route planning in dynamic multi-hop ridesharing. *Proceedings of the 11th European conference on Evolutionary computation in combinatorial optimization, Torino, Italy*, 84–95.

Herbawi, W. and M. Weber (2012). A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. *In Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*.

Hosni, H., J. Naoum-Sawaya, and H. Artail (2014). The shared-taxi problem: Formulation and solution methods. *Transportation Research Part B: Methodological 70*, 303–318.

Jaw, J.-J., A. R. Odoni, H. N. Psaraftis, and N. H. Wilson (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological 20*(3), 243 – 257.

k. Naughton (2015). Driverless, shared cars to cut vehicle ownership by half: Barclays. Accessed Oct. 6, 2015.

Kleiner, A., B. Nebel, and V. Ziparo (2011). A mechanism for dynamic ride sharing based on parallel auctions.

Kowshik, R., J. Gard, J. Loo, P. P. Jovanis, and R. Kitamura (1993). Development of user needs and functional requirements for a real-time ridesharing system. *California Partners for Advanced Transit and Highways (PATH).*

Levofsky, A. and A. Greenberg (2001). Organized dynamic ride sharing: The potential environmental benefits and the opportunity for advancing the concept. In *Transportation Research Board 2001 Annual Meeting*, pp. 7–11.

Li, X. and L. Quadrifoglio (2010). Feeder transit services: choosing between fixed and demand responsive policy. *Transportation Research Part C: Emerging Technologies 18*(5), 770–780.

Liaw, C.-F., C. C. White, and J. Bander (1996). A decision support system for the bimodal dial-a-ride problem. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 26*(5), 552–565.

Mas-Colell, A., M. D. Whinston, J. R. Green, et al. (1995). *Microeconomic theory*, Volume 1. Oxford university press New York.

Masson, R., F. Lehuédé, and O. Péton (2014a). The dial-a-ride problem with transfers. *Computers & Operations Research 41*(0), 12 – 23.

Masson, R., A. Trentini, F. Lehuédé, N. Malhéné, O. Péton, and H. Tlahig (2014b). Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, 1–29.

Myerson, R. B. and M. A. Satterthwaite (1983). Efficient mechanisms for bilateral trading. *Journal of economic theory 29*(2), 265–281.

Nelson Nygaard Consulting Associates and RideNow Inc (2006). Ridenow evaluation draft final report. In *Prepared for the Alameda County Congestion Management Agency.*

Nisan, N. and A. Ronen (1999). Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 129–140. ACM.

O'Sullivan, S. (2011). Case study in real-time ridesharing: Sr 520 carpooling pilot project, seattle, wa. In *18th ITS World Congress.*

Psaraftis, H. N. (1983). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science 17*(3), 351–357.

Qiu, F., W. Li, and J. Zhang (2014). A dynamic station strategy to improve the performance of flex-route transit services. *Transportation Research Part C: Emerging Technologies 48*, 229–240.

Quadrifoglio, L., M. M. Dessouky, and F. Ordóñez (2008). Mobility allowance shuttle transit (mast) services: Mip formulation and strengthening with logic constraints. *European Journal of Operational Research 185*(2), 481–494.

Rabadi, G., R. J. Moraga, and A. Al-Salem (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing 17*(1), 85–97.

RTrip (2013). Avego go520 is new and improved. `http://www.gortripblog.com/2011/07/avego-go520-is-new-and-improved/`. Accessed: April 2013.

Savelsbergh, M. W. P. and M. Sol (1995). The general pickup and delivery problem. *Transportation Science 29*(1), 17–29.

Schall, E. (2015). The future of car ownership and autonomous driving. Accessed Aug. 1 2015.

Schaub, T., G. Friedrich, and B. O'Sullivan (2014). *ECAI 2014: 21st European Conference on Artificial Intelligence*, Volume 263. IOS Press.

Schaub, T., F. Gerhard, and O. Barry (2014). *ECAI 2014: 21st European Conference on Artificial Intelligence*, Volume 263. IOS Press.

Schoettle, B. and M. Sivak (2015). Potential impact of self-driving vehicles on household vehicle demand and usage.

Shaheen, S. and N. Chan (2015). Mobility and the sharing economy: Impacts Synopsis. Technical report, TRANSPORTATION SUSTAINABILITY RESEARCH CENTER, University of California Berkeley.

Stein, D. M. (1978). Scheduling dial-a-ride transportation systems. *Transportation Science 12*(3), 232–249.

Steinbach, M., G. Karypis, V. Kumar, et al. (2000). A comparison of document clustering techniques. In *KDD workshop on text mining*, Volume 400, pp. 525–526. Boston.

Stiglic, M., N. Agatz, M. Savelsbergh, and M. Gradisar (2015). The benefits of meeting points in ride-sharing systems. *ERIM Report Series Reference No. ERS-2015-003-LIS*.

Teodorović, D. and M. Dell'Orco (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. In *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*, pp. 51–60.

Tientrakool, P. (2011). *Reliable Neighborcast Protocol for Vehicular Ad hoc Networks*. Ph. D. thesis, Columbia University.

UCC (2013). Avego to launch world's first commuter service in cork. `http://collegenews.ie/index.php/1601/express/express-news/avego-to-launch-world-first-commuter-service-in-cork/`. Accessed: April 2013.

U.S. Department of Transportation, F. H. A. (2009). Ridesharing in north america: Past, present, and future. pp. 2.

Wang, X. (2013). Optimizing ride matches for dynamic ride-sharing systems.

Wego rideshare (2013). Avego go520 is new and improved. `http://wegorideshare.com/`. Accessed: April 2013.

Wolfler Calvo, R., F. de Luigi, P. Haastrup, and V. Maniezzo (2004). A distributed geographic information system for the daily car pooling problem. *Computers & Operations Research 31*(13), 2263 – 2278.

Yamashita, T. (2015). Implementation in weakly undominated strategies: Optimality of second-price auction and posted-price mechanism. *The Review of Economic Studies*, rdv018.

Appendices

A Proof of Proposition

Proposition 2. *Number of connections for rider r can be calculated using term $\sum_{d \in D} u_r^d - 1$.*

Proof. If driver d carries rider r on any link, then $u_r^d = 1$. So $\sum_{d \in D} u_r^d - 1$ yields the number of connections, only if a driver does not pick up a rider multiple times. Without loss of generality, we use the example in Figure A.1 to show by contradiction that such a situation cannot happen. Figure A.1 shows a rider's itinerary. On the first and third links, the rider is traveling with d_1, and on the second link, he/she is traveling with d_2. If d_1 travels on both ℓ_1 and ℓ_3, at some point d_1 must have traveled from node 2 to node 3, and the rider could have accompanied him on that ride too, reducing the number of connections from 2 to 0. The term $-W_r \sum u_r^d$ in the maximization objective function ensures that the route with zero connections is selected as the optimal solution. Alternatively, one could leave out this term from the objective function, and use post-processing to refine the solution. □

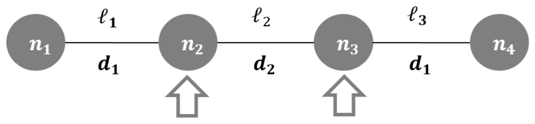


Figure A.1: An undesirable match

B Proof of Proposition

Proposition 3. *Constraint sets (4.5k) and (4.5l) register all drivers who collectively form each rider's itinerary.*

Proof. From constraint set (4.5k) we have $u_r^d \geq y_\ell^{rd}$. If $y_\ell^{rd} = 1$, u_r^d is forced to be 1. If $y_l^{rd} = 0$, u_r^d can take either 0 or 1. Constraint set (4.5l) ensures that in the latter case, u_r^d takes its lower bound. $\square$

C Decomposition Algorithm

We start the algorithm by setting the iteration counter (i) to 1, and the sub-problem counter (k) to 0. Each rider forms a new sub-problem in the first iteration of the algorithm, and the type of all sub-problems is set as active. In Algorithm C, R_i^k denotes the set of riders in sub-problem k of iteration i, and SP_i denotes the set of sub-problems in iteration i. $Active(k)$ indicates whether the type of sub-problem k is active (1) or not (0).

The second step of the algorithm involves solving the set of active sub-problems in the current iteration. We keep in set D_i^k all matched drivers in sub-problem k of iteration i, and in set D_i all drivers who have been matched in iteration i. Furthermore, we use sets $Y_{i,k}$, $X_{i,k}^d$, and $Z_{i,k}^r$ to keep a track of the itineraries of all riders, driver d ($\forall d \in D$), and whether rider r ($\forall r \in R$) has been matched in sub-problem k of iteration i, respectively.

In step 3 we check the stopping criterion of the algorithm. If each driver receives the same itinerary under all sub-problems, then the union of solutions to the sub-problems will yield the optimal solution to the original optimization problem (X^*).

Algorithm C The decomposition algorithm

```
Step 1. Initialize
$i \leftarrow 1$
$k \leftarrow 0$
For each rider $r \in R$
    $k \leftarrow k + 1$.
    Set $R_i^k = \{r\}$.
    Set $SP_i = \{1, ..., |R|\}$.
    Set $Active(k) \leftarrow 1$.
End For
Step 2. Solve the (active) sub-problems
For each sub-problem $k \in SP_i$ for which $Active(k) = 1$
    Solve sub-problem $k$.
    Let $D_i^k = \{d \in D \mid \sum_{r \in R_i^k, \ell \in L_{rd}} y_\ell^{rd*} \geq 1\}$.
    Let $D_i = \bigcup_{k \in SP_i} D_i^k$.
    Let $Y_{i,k} = \left(y_\ell^{rd*}\right)_{r \in R_i^k, d \in D_i^k, \ell \in L_{rd}}$.
    Let $X_{i,k}^d = \left(x_\ell^{d*}\right)_{\ell \in L_d}$, $\forall d \in D_i^k$.
    Let $Z_{i,k}^r = z_r^*$, $\forall r \in R_i^k$.
End For
Step 3. Termination Criterion
For each driver $d \in D_i$
    $Terminate(d) \leftarrow 0$.
    If $X_{i,m}^d = X_{i,n}^d$ for all $(m, n) \in D_i^m \times D_i^n$
        $Terminate(d) \leftarrow 1$.
    End IF
End For
If $\sum_{d \in D_i} Terminate(d) = |D|$ then
    Stop.
    $X^* = U_{k \in SP_i}\left(Y_{i,j}, \cup_{d \in D} X_{i,k}^d, \cup_{r \in R} Z_{i,k}^r\right)$.
Else
    Continue to Step 4.
End If
```

If the algorithm does not terminate in step 3, we move forward to step 4 to form the set of sub-problems for the next iteration. We start step 4 by defining set R' to include all riders that need to be allocated to sub-problems in the new iteration, and initialize it with R. Next, we go through the list of matched drivers in the previous iteration (in no particular order). For each driver, we find the riders in set R' that have the driver on their itinerary. These riders (if belonging to different sub-problems) will from a new sub-problem in the current iteration, and are removed from set R'. We mark all such sub-problems k as applicable by setting $Applicable(k) \leftarrow 1$.

At this point, there could still be riders that do not belong to any sub-problems in the new iteration. Next, we go through the sub-problems from the previous iteration one by one. For a given sub-problem j from the previous iteration, we find all riders r who do not belong to any sub-problems in the current iteration (i.e. $r \in R'$). If such riders exist, they will from a new sub-problem in the current iteration. If it turns out that we are importing an entire sub-problem from the previous iteration to the current iteration, then this sub-problem does not need to be solved again, and so we set the applicability indicator of the sub-problem to zero. Otherwise, if we are importing only a part of a sub-problem from the previous iteration as a new sub-problem to the current iteration, then the solution from the previous iteration is not necessarily optimal anymore, and so we set the applicability indicator of the new sub-problem to 1.

Algorithm C (continued) The decomposition algorithm

```
Step 4. Form the set of sub-problems for the next iteration
$i \leftarrow i+1$
$k \leftarrow 0$
Let $R' = R$.
For each driver $d \in D_{i-1}$ such that there are at least two sub-problems $m, n \in SP_{i-1}$ with $d \in D^m_{i-1}$ and $d \in D^n_{i-1}$
    Let $R_{temp}$ = set of all riders $r \in R'$ that have driver $d$ on their optimal path
    in any sub-problems $k \in SP_{i-1}$.
    If $R_{temp} \neq \emptyset$ then
        Form a new sub-problem: $k \leftarrow k+1$.
        Set $R^k_i = R_{temp}$.
        Update $R' \leftarrow R' \backslash R_{temp}$.
        Set $Applicable(k) \leftarrow 1$.
    End If
End For
For $j \in SP_{i-1}$
    If $R^j_{i-1} \cap R' \neq \emptyset$ and $|R^j_{i-1} \cap R'| < |R^j_{i-1}|$ then
        Form a new sub-problem: $k \leftarrow k+1$.
        Set $R^k_i = R^j_{i-1} \cap R'$.
        Set $Applicable(k) \leftarrow 1$.
    Elseif $|R^j_{i-1} \cap R'| = |R^j_{i-1}|$
        Form a new sub-problem: $k \leftarrow k+1$.
        Set $R^k_i = R^j_{i-1}$.
        Set $Applicable(k) \leftarrow 0$.
    End If
End For
```

In step 5, we prevent the algorithm from looping between iterations. After forming the set of sub-problems for the current iteration, we compare these sub-problems with sub-problems from previous iterations. If it turns out that two iterations have the exact same set of sub-problems, we form a new intermediate sub-problem in the current iteration by finding two sub-problems from the previous iteration such that each of these two sub-problems includes a subset of riders in a sub-problem in the current iteration. These two sub-problems form a new sub-problem in the current iteration. Naturally, riders in this newly formed sub-problem are eliminated from their original sub-problems, and the newly formed intermediate sub-problem is marked as applicable.

The last step of the algorithm is to find the active sub-problems: those whose solutions cannot be derived from the solutions of previously solved sup-problems. There are two cases where sub-problems are not active: first, if the exact same sub-problem has been solved before, in which case the solution is readily available; second, if a sub-problem k is a union of a set of sub-problems from a previous iteration, and the solutions to these sub-problems do not have any conflicts in terms of itineraries of drivers. In this case, the solution to sub-problem k would be the union of the solutions to this set of sub-problems.

Algorithm C (continued) The decomposition algorithm

```
Step 5. Check for repeating patterns and form intermediate sub-problems
For $j \in 1$ to $i-2$
    If iterations $i$ and $j$ have the same set of sub-problems, then
        For sub-problem $k \in SP_i$
            Find two sub-problems $m, n \in SP_{i-1}$ such that they each share
            at least one rider with $R_i^k$.
            Delete all riders in $R_i^m \cup R_i^n$ from sub-problems in $SP_i$.
            Add a sub-problem $k$ to $SP_i$ with combined set of riders $R_i^m \cup R_i^n$.
            Set $Applicable(k) \leftarrow 1$.
            Exit For loop.
        End For
    End If
End For
Step 6. Identify active sub-problems
For each sub-problem $k \in SP_i$ such that $Applicable(k) = 1$
    Set $Active(k) \leftarrow 1$
    For each iteration $j = 1$ to $i-1$
        For each sub-problem $n \in SP_j$
            While $Active(k) = 1$
                If $R_i^k = R_j^n$ then
                    The solution to $k$ is already available:
                    Set $Y_{i,k} = Y_{j,n}$.
                    Set $X_{i,k}^d = X_{j,n}^d \ \forall d \in D_j^n$.
                    Set $Z_{i,k}^r = Z_{j,n}^r \ \forall r \in R_n^j$.
                    Set $Active(k) \leftarrow 0$.
                Elseif there exist a set of sub-problems $N \in SP_j$ such that
                $R_i^k = \cup_{n \in N} R_j^n$, and the solutions to sub-problems in set $N$
                have no conflict (i.e., $\forall m, n \in N$ and $\forall d \in D_j^m \cap D_j^n \ : \ X_{j,m}^d = X_{j,n}^d$ )
                    The solution to $k$ is the union of the solutions to $n \in N$:
                    Set $Y_{i,k} = \left(Y_{j,n}\right)_{n \in N}$.
                    Set $X_{i,k}^d = X_{j,n}^d \ \forall d \in \bigcup_{n \in N} D_j^n$.
                    Set $Z_{i,k}^r = Z_{j,n}^r \ \forall r \in \bigcup_{n \in N} R_n^j$.
                    Set $Active(k) \leftarrow 0$.
                End If
            End While
        End For
    End For
End For
Go to Step 2
```

D Revised version of the P2P multi-hop matching problem

Each sub-problem k in iteration i of the decomposition algorithm needs to be solved using the optimization problem in (D.1). In the interest of simplicity of notation, let us denote R_i^k and D_i^k, i.e. the rider and drivers sets in sub-problem k in iteration i, by R_κ and D_κ respectively.

$$\text{Max} \quad \sum_{r \in R_\kappa} z_r \quad - \sum_{r \in R_\kappa} W_r \sum_{d \in D_\kappa : (r,d) \in M} u_r^d \tag{D.1a}$$

$$\sum_{\substack{\ell \in L_d: \\ s_i = OS_d}} x_\ell^d \quad - \sum_{\substack{\ell \in L_d: \\ s_j = OS_d}} x_\ell^d = 1;\ \forall d \in D_\kappa \tag{D.1b}$$

$$\sum_{\substack{\ell \in L_d: \\ s_j = DS_d}} x_\ell^d \quad - \sum_{\substack{\ell \in L_d: \\ s_i = DS_d}} x_\ell^d = 1;\ \forall d \in D_\kappa \tag{D.1c}$$

$$\sum_{\substack{t_i, s_i \\ \ell = (t_i, s_i, t, s) \in L_d}} x_\ell^d \quad = \sum_{\substack{t_j, s_j \\ \ell = (t, s, t_j, s_j) \in L_d}} x_\ell^d;\ \forall d \in D_\kappa, \forall t \in T_d, \forall s \in G_d \backslash \{OS_d \cup DS_d\} \tag{D.1d}$$

$$\sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} \sum_{\substack{\ell \in L_{rd}: \\ s_i = OS_r}} y_\ell^{rd} - \sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} \sum_{\substack{\ell \in L_{rd}: \\ s_j = OS_r}} y_\ell^{rd} = z_r;\ \forall r \in R_\kappa \tag{D.1e}$$

$$\sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} \sum_{\substack{\ell \in L_{rd}: \\ s_j = DS_r}} y_\ell^{rd} - \sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} \sum_{\substack{\ell \in L_{rd}: \\ s_i = DS_r}} y_\ell^{rd} = z_r;\ \forall r \in R_\kappa \tag{D.1f}$$

$$\sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} \sum_{\substack{t_i, s_i: \\ \ell = (t_i, s_i, t, s) \in L}} y_\ell^{rd} = \sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} \sum_{\substack{t_j, s_j: \\ \ell = (t, s, t_j, s_j) \in L}} y_\ell^{rd};\ \forall r \in R_\kappa, \forall t \in T_r, \forall s \in G_r \backslash \{OS_r \cup DS_r\} \tag{D.1g}$$

$$\sum_{\substack{r \in R_\kappa: \\ (r,d) \in M, \ell \in L_{rd}}} y_\ell^{rd} \le C_d x_\ell^d;\ \forall d \in D_\kappa, \forall \ell \in L_d \tag{D.1h}$$

$$u_r^d \ge y_\ell^{rd};\ \forall (r,d) \in M, \forall \ell \in L_{rd} \tag{D.1i}$$

$$u_r^d \le \sum_{\ell \in L_{rd}} y_\ell^{rd};\ \forall (r,d) \in M \tag{D.1j}$$

$$\sum_{\substack{d \in D'_\kappa: \\ (r,d) \in M}} y_\ell^{rd} - 1 \le V_r;\ \forall r \in R_\kappa \tag{D.1k}$$