Real-Time Optimization for Dynamic Ride-Sharing

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Martin Pouls, M.Sc.

Tag der mündlichen Prüfung:	05.05.2023
Erster Gutachter:	Prof. Dr. Stefan Nickel
Zweiter Gutachter:	Univ.-Prof. Dr. Jan Fabian Ehmke

Karlsruhe, 2023

Abstract

Throughout the last decade, the advent of novel mobility services such as ride-hailing, car-sharing, and ride-sharing has shaped urban mobility. While these types of services offer flexible on-demand transportation for customers, they may also increase the load on the, already strained, road infrastructure and exacerbate traffic congestion problems. One potential way to remedy this problem is the increased usage of dynamic ride-sharing services. In this type of service, multiple customer trips are combined into share a vehicle simultaneously. This leads to more efficient vehicle utilization, reduced prices for customers, and less traffic congestion at the cost of slight delays compared to direct transportation in ride-hailing services.

In this thesis, we consider the planning and operation of such dynamic ride-sharing services. We present a wider look at the planning context of dynamic ride-sharing and discuss planning problems on the strategical, tactical, and operational level. Subsequently, our focus is on two operational planning problems: dynamic vehicle routing, and idle vehicle repositioning.

Regarding vehicle routing, we introduce the vehicle routing problem for dynamic ride-sharing and present a solution procedure. Our algorithmic approach consists of two phases: a fast insertion heuristic, and a local search improvement phase. The former handles incoming trip requests and quickly assigns them to suitable vehicles while the latter is responsible for continuously improving the current routing plan. This way, we enable fast response times for customers while simultaneously effectively utilizing available computational resources.

Concerning the idle vehicle repositioning problem, we propose a mathematical model that takes repositioning decisions and adequately reflects available vehicle resources as well as a forecast of the upcoming trip request demand. This model is embedded into a real-time planning algorithm that regularly re-optimizes the movement of idle vehicles. Through an adaptive parameter calculation process, our algorithm dynamically adapts to changes in the current system state.

To evaluate our algorithms, we present a modular simulation-based evaluation framework. We envision that this framework may also be used by other researchers and developers. In this thesis, we perform computational evaluations on a variety of scenarios based on real-world data from Chengdu, New York City, and Hamburg. The computational results show that we are able to produce high-quality solutions in real-time, enabling the usage in high-demand settings. In addition, our algorithms perform robustly in a variety of settings and are quickly adapted to new application settings, such as the deployment in a new city.

Contents

List of Figures

List of Tables

Glossary

ADP	approximate dynamic programming
AGV	autonomous guided vehicle
ALNS	adaptive large neighborhood search
BEV	battery-electric vehicle
CNN	convolutional neural network
DARP	dial-a-ride problem
DDARP	dynamic dial-a-ride problem
DIS	dispatching algorithm
DP	dynamic programming
DVRPSC	dynamic vehicle routing problem with stochastic customers
FDR	forecast-driven repositioning algorithm
FDR-M	forecast-driven repositioning MIP model
GPS	global positioning system
LA	lookahead algorithm
LS	local search algorithm
LSTM	long short-term memory
MaaS	mobility-as-a-service
MIP	mixed-integer programming
MOD	mobility-on-demand
MSA	multiple scenario approach
OSM	OpenStreetMap
PDP	pickup and delivery problem
PFA	policy function approximation
PT	public transport

REACT	reactive repositioning algorithm
RL	reinforcement learning
RMSE	root-mean-square error
SimDRS	simulation of dynamic ride-sharing
TS	tabu search
VFA	value function approximation
VRP	vehicle routing problem
VRPDRS	vehicle routing problem for dynamic ride-sharing

1 Introduction

Urban mobility has undergone significant changes throughout the last decade. Due to the widespread availability of modern communication technology and smartphone applications, novel mobility services such as ride-hailing, bike- or scooter-sharing, and ride-sharing have appeared. Previously, few transportation modes, most importantly private cars, taxis, and public transport, dominated the market of urban mobility. Nowadays, mobility service providers such as Uber, Lyft or Tier offer on-demand transportation in urban areas, sometimes even combining multiple modes of transportation into integrated trips.

However, the increased convenience for travelers may come at a cost regarding traffic congestion and the environmental impact through pollution and greenhouse gas emissions. Early reports and studies on the impact of ride-hailing services indicate that the availability of ride-hailing can lead to additional road traffic as customers replace public transport voyages with ride-hailing trips. Additionally, customers may undertake trips that they would otherwise have done without. Observations from New York City (Fitzsimmons 2017) suggest that the recent decline in subway ridership might be attributed to the replacement by ride-hailing trips with Uber and Lyft. Similarly, surveys among ride-hailing users indicate that up to 34 % of travelers would have otherwise used public transport or active transportation modes such as walking or biking (Gehrke et al. 2019). While the exact relationship and interaction of modern mobility services with public transport and private car ownership is still an active research topic (Nelson and Sadowsky 2019; Cats et al. 2022), results such as the aforementioned suggest that ride-hailing, in particular, may lead to increased traffic congestion and emissions in cities through the substitution of public transport, walking and biking trips. On the other hand, some studies also identify the potential for mobility-on-demand (MOD) services to complement existing public transport options by forming multi-modal trips (Gehrke et al. 2019). This way, mobility-on-demand may increase public transport utilization and reduce traffic congestion problems.

Another option for making mobility-on-demand more efficient, is the increased usage of shared mobility options. Ride-sharing services, in contrast to ride-hailing, combine multiple customers to single trips with multiple passenger groups sharing a vehicle simultaneously. This leads to more efficient vehicle usage and may remedy the impact on traffic and the environment.

Planning and operating such ride-sharing services comes with a key set of challenges. Service providers operate in a highly dynamic environment with customers demanding timely service and quick response times to their trip requests. Moreover, operators must be equipped to handle thousands of trip requests per hour. These requirements raise the necessity for tailor-made operational planning algorithms, particularly for planning vehicle movements. In addition, to assess the performance of these algorithms under realistic scenarios, we need simulation-based evaluation tools. Such simulation tools may also be utilized by service providers to adequately plan the setup of a ride-sharing service before deploying it in a new city.

1.1 Scope and Contribution of this Thesis

In this thesis, we consider precisely this application setting of dynamic ride-sharing as described above. We discuss the particular challenges arising from ride-sharing, such as the high degree of dynamism and the high request load, in detail, and present insights into the different stakeholders, objectives, and planning problems. Regarding the operational planning of ride-sharing services, we propose our own specialized solution algorithms and a simulation-based evaluation framework to evaluate these algorithmic approaches under realistic simulation scenarios. The main contributions of our work can be summarized as follows.

We take a detailed look at the planning context of ride-sharing services. As a first step, we **compare ride-sharing to other mobility-as-a-service (MaaS) applications** and present aspects that differentiate ridesharing from mobility services such as ride-hailing or car-sharing. Subsequently, we discuss characteristics, business models and planning objectives for different stakeholders of dynamic ride-sharing systems. Based on these insights, we **derive a set of strategical, tactical and operational planning problems that are relevant for ride-sharing operators** and illustrate how the planning level may interact with the actual execution of shared rides.

Subsequently, we focus on **two key operational planning problems in the context of ride-sharing: dynamic vehicle routing and idle vehicle repositioning**. In the vehicle routing problem, the objective is to plan efficient vehicle routes that fulfill customer requests while respecting a set of relevant constraints such as vehicle capacities and customer time windows. Additionally, the vehicle routing algorithm must be able to process high request volumes in real-time to enable fast response times to customer requests. For this purpose, we propose a solution approach that combines a fast insertion heuristic with a local search improvement phase. This allows us to quickly respond to new trip requests while at the same time utilizing the available computational power to improve the routing plan. Besides vehicle routing, we also consider idle vehicle repositioning. This is a problem that arises in a ride-sharing setting whenever there is a spatial imbalance between the vehicle supply and the anticipated trip request demand. The objective is to reposition available vehicles in a way that they are well-suited to serve future customers. For this purpose, we propose a mathematical model that is integrated into our real-time planning process. Our model reflects a forecast of the upcoming demand and the current system state and capabilities of vehicles to serve future trip requests. In addition, it considers the cost and operational overhead of repositioning movements. In the design of our operational planning algorithms we aim to consider practical requirements for ride-sharing providers. For instance, the algorithms are equipped to handle high trip request loads in real-time and are easily transferred to new application settings, for instance, a new city in which the provider operates.

These planning algorithms are integrated into a **modular simulation-based evaluation framework**. We envision that this framework may be used by algorithm developers to assess the performance of their optimization algorithms based on simulation scenarios. In this thesis, we utilize our framework to evaluate our vehicle routing and repositioning approaches on a **diverse set of simulation scenarios derived from real-world data from multiple cities**. For instance, we run simulations with multiple fleet sizes, vehicle capacities, and varying demand patterns on different weekdays. Such simulation scenarios can yield insights into potential practical applications of ride-sharing services, for example, when a service operator plans to expand to a new city or region.

The work presented in this thesis has been published in three scientific papers:

1. M. Pouls et al. (2020). Idle Vehicle Repositioning for Dynamic Ride-Sharing. *Computational Logistics.* Ed. by E. Lalla-Ruiz et al. Vol. 12433. Cham: Springer International Publishing, pp. 507–521. DOI: `10.1007/978-3-030-59747-4_33`.
2. M. Pouls et al. (2021). Real-Time Dispatching with Local Search Improvement for Dynamic Ride-Sharing. *Computational Logistics.* Ed. by M. Mes et al. Vol. 13004. Cham: Springer International Publishing, pp. 299–315. DOI: `10.1007/978-3-030-87672-2_20`.
3. M. Pouls et al. (2022). Adaptive forecast-driven repositioning for dynamic ride-sharing. *Annals of Operations Research.* DOI: `10.1007/s10479-022-04560-3`.

1.2 Organization

The structure of this thesis is illustrated in Figure 1.1.

Chapter 2 defines the term mobility-as-a-service as it is used in this thesis and presents a taxonomy of different MaaS applications including ride-sharing. These different applications are compared regarding several aspects concerning their modes of operation.

In Chapter 3, we address dynamic ride-sharing in detail. We discuss different business models, stakeholders, and objectives. Subsequently, we present an overview of strategical, tactical, and operational planning problems that arise when operating a ride-sharing service.

Chapter 4 introduces our simulation-based evaluation framework. We compare it to related simulation approaches and explain our design decisions. Additionally, we detail the different elements, input data, and communication between components.

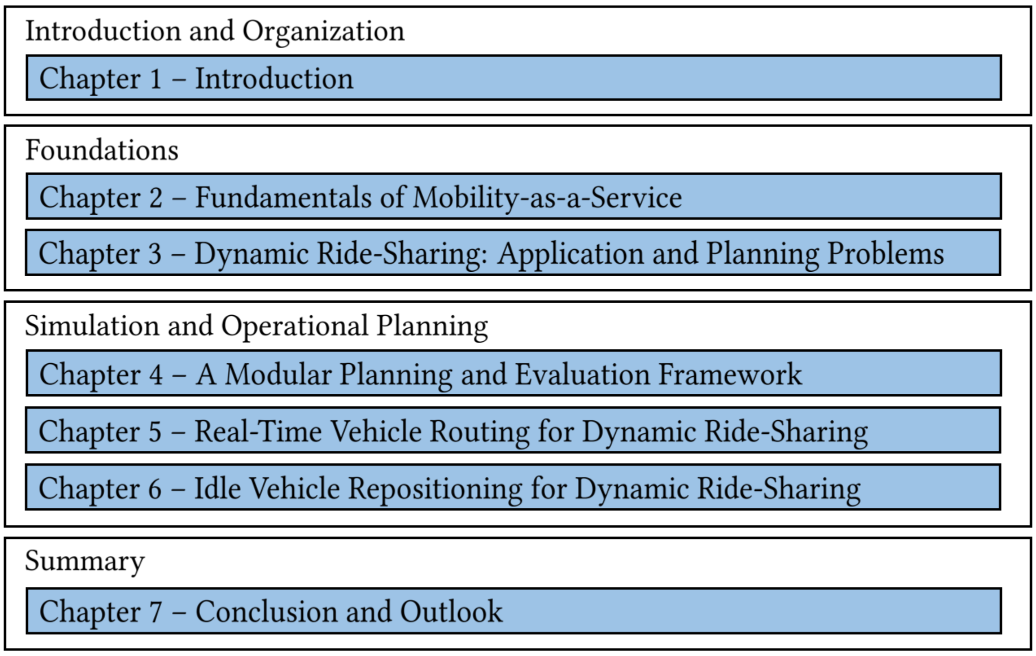

Introduction and Organization
Chapter 1 – Introduction
Foundations
Chapter 2 – Fundamentals of Mobility-as-a-Service
Chapter 3 – Dynamic Ride-Sharing: Application and Planning Problems
Simulation and Operational Planning
Chapter 4 – A Modular Planning and Evaluation Framework
Chapter 5 – Real-Time Vehicle Routing for Dynamic Ride-Sharing
Chapter 6 – Idle Vehicle Repositioning for Dynamic Ride-Sharing
Summary
Chapter 7 – Conclusion and Outlook

Figure 1.1: Structure of this thesis.

In Chapters 5 and 6, we propose our solution algorithms for the operational vehicle routing and idle vehicle repositioning problems in the context of dynamic ride-sharing. We review related approaches in these fields and present our own solution approaches in detail. Both algorithms are evaluated utilizing our simulation framework on real-world data from the cities of Chengdu, Hamburg, and New York City.

Finally, we summarize our findings and contributions in Chapter 7 and propose potential directions for future research.

2 Fundamentals of Mobility-as-a-Service

In this chapter, we provide an introduction into the field of mobility-as-a-service (MaaS). We first define the term MaaS as it is used in this thesis. Subsequently, we present a taxonomy of different mobility services and dicuss and categorize them based on a set of service characteristics. We pay particular attention to the differences between ride-sharing, the focus of this thesis, and other mobilty services.

2.1 A Definition of Mobility-as-a-Service

While the term mobility-as-a-service is relatively new, MaaS applications in the broadest sense have existed for decades. Some newer publications present a relatively narrow definition of MaaS in which they consider the intermodal integration of different transportation services as an integral aspect (Utriainen and Pöllänen 2018; Jittrapirom et al. 2017). In this work, we adopt a broader meaning of MaaS that encompasses all mobility services in which the means of transportation are not solely used by the owner as is generally the case with privately owned cars. Instead, transportation services are also offered to other customers. The term mobility-on-demand is sometimes used interchangeably with mobility-as-a-service. We consider mobility-on-demand to be a subset of mobility-as-a-service in which services are offered on-demand, i.e. dynamically based on specific transportation requests by customers. This is in contrast to services that are mostly bound by timetables such as public transport or services like traditional car-pooling in which transportation opportunities are based on the available routes.

2.2 A Taxonomy of Mobility-as-a-Service Applications

As the difference between several mobility services such as car-sharing, ride-hailing and ride-sharing is not clearly defined in literature and the terms are sometimes used interchangeably, we present a definition and taxonomy of different mobility services. It is our goal to encompass the most important MaaS applications. However, due to the influx of new mobility offerings and business models in recent years, it is not within the scope of this work to present a complete overview of all possible forms of mobility services. We differentiate services based on the following criteria and summarize our results in Table 2.1.

Stations We distinguish between services that use *physical* or *virtual* stations or *none* at all. In station-based models, transportation is only offered between predefined stations. This allows for a simplified planning process and consolidation of demand at stations. In contrast, many modern MaaS solutions are not bound to stations anymore and are therefore able to offer a more flexible service. An intermediate solution between the physical station infrastructure of public transport solutions and the complete absence of stations is the usage of virtual stations. This concept has arisen in some recent transportation offerings such as ride-sharing or scooter-sharing. For instance, in the case of scooter sharing, virtual stations may be displayed in a mobile application and designate areas in which scooters may be dropped off. However, no physical infrastructure exists at these locations which makes it easy to add or remove virtual stations.

Timetable While most traditional public transportation models are bound by a predefined timetable, this is no longer the case for newer MaaS applications in which the service is provided on-demand. In timetable-based services, vehicles operate according to a fixed schedule and customers must plan their trips in accordance with this timetable. In contrast, on-demand mobility services offer flexible transportation where customers may request a trip at any point in time and are served in a timely manner.

Vehicle sharing We differentiate between services in which vehicles may be shared between multiple customers simultaneously compared to services in which a customer books the vehicle exclusively for themselves. The former generally offer lower prices while the latter offer increased comfort and flexibility for the customer.

Vehicle ownership Services may be classified according to the ownership of the vehicles. Either vehicles are owned by the *operator* of the service or by other *private* entities who then offer their vehicles via a platform provider. For some use cases, such as ride-sharing, both business models are possible.

Driver employment Connected to the previous aspect is the employment of drivers. These may either be *employed* with the service operator or work *independently* and offer their services via a platform. We differentiate vehicle and driver ownership as they are not always correlated. For instance, it is a possible business model to have drivers who are employed with a service operator but utilize their private vehicles to provide transportation services. In some services, such as car-sharing, there are no dedicated drivers and the vehicle is *customer-operated.*

In the following we briefly discuss all MaaS applications summarized in Table 2.1. We provide an overview of the key aspects of all services and, in particular, discuss the differences compared to ride-sharing services, which are the focus of this thesis. We also provide references to more in-depth reviews concerning each service type.

	Stations	**Timetable**	**Vehicle sharing**	**Vehicle ownership**	**Driver employment**	**References**
Public transport	physical	yes	yes	operator	employed	Desaulniers and Hickman (2007) Ibarra-Rojas et al. (2015)
Taxi and ride-hailing	physical / virtual / none	no	no	both	both	Salanova et al. (2011) Tirachini (2020)
Car-pooling	none	no	yes	private	independent	Agatz et al. (2012) Shaheen and Cohen (2019) Tafreshian et al. (2020)
Car-sharing	physical / none	no	no	operator	customer-operated	Jorge and Correia (2013) Ferrero et al. (2018) Huang et al. (2020b) Golalikhani et al. (2021)
Bike- and scooter-sharing	physical / virtual / none	no	no	operator	customer-operated	Shaheen et al. (2010) Fishman et al. (2013) Shen et al. (2018)
Ride-sharing	virtual / none	no	yes	both	both	Shaheen and Cohen (2019)

Table 2.1: A taxonomy of mobility-as-a-service applications.

Public transport Public transport (PT) is among the oldest MaaS applications, with its beginnings dating back to the 17th century (American Public Transportation Association 2007, p. 5). Compared to more modern services, PT offers the least amount of flexibility due to its mostly station-based and timetable-bound nature. More recently, PT providers have started to offer on-demand bus services. However, these are still relatively fringe concepts and the vast majority of public transport is still based on predefined bus or railway lines and timetables. Due to the nature of these services, public transport comes with a set of planning problems that differ from most other MaaS applications considered in this section. For instance, a key central problem is the determination of timetables and subsequently the assignment of drivers and vehicles to specific routes. This combination of planning problems is unique to timetable- and station-based services. Hence, based on these characteristics, we see limited overlap between PT and dynamic ride-sharing. There is some common ground in strategical infrastructure planning (e.g. vehicle depots) and personnel scheduling. However, the operational planning problems are vastly different. For comprehensive reviews concerning public transportation, we refer the readers to Desaulniers and Hickman (2007) and Ibarra-Rojas et al. (2015).

Taxi and ride-hailing While taxis have long been a staple mobility service, more recently the term ride-hailing has emerged to describe similar app-based services such as Uber and Lyft. We group these applications together as the offered service is very similar and consequently the associated planning problems are closely related. Both taxi and ride-hailing providers offer direct transportation from one location to another while transporting only one customer group at a time. These types of services are generally not timetable- or station-based although there may sometimes be designated pickup and waiting areas (e.g. at airports) or virtual stops. Several business models are common regarding vehicle ownership and driver employment. Service providers such as MOIA may own the vehicle fleet and use employed drivers to provide their service. On the other hand, intermediary platforms such as Uber or Lyft merely act as a broker between customers and drivers while the latter are independent and use their private cars. Operational planning for these services is comparatively simple as no complex vehicle routing algorithms are necessary. This is due to the fact that only one trip request is served at a time. In contrast, in ride-sharing multiple customer groups may occupy the vehicle simultaneously, leading to a more complex routing problem. However, the idle vehicle repositioning problem arising from taxi and ride-hailing applications is similar to ride-sharing and may lead to approaches that can be applied to both domains. This aspect is discussed in more detail in Chapter 6. Detailed reviews on taxi and ride-hailing services may be found in Salanova et al. (2011) and Tirachini (2020).

Car-pooling Car pooling, at times also referred to as *social* ride-sharing or *peer-to-peer* ride-sharing, describes a service in which private car owners offer to pick up additional passengers along their planned route. Some literature may also use the term ride-sharing to describe such services. However, in this work, we exclusively use ride-sharing for services in which transportation is offered as a main dedicated service, whereas in car-pooling the opportunity to transport other travelers arises as a by-product. One common use case of car-pooling is commuting to work and offering to transport additional passengers with a similar itinerary. The earliest applications of car-pooling date back to World War II and the energy crisis in the 1970s when car-pooling was used as a means of saving rubber and fuel (Chan and Shaheen 2012). More recently, car-pooling has become increasingly popular due to the ease of dynamically matching riders via mobile applications. As mentioned above, the key difference compared to ride-sharing is the fact that in car-pooling the objective is to match customers to vehicles with an existing route, for instance, a commute to work. In contrast, in ride-sharing, there are dedicated drivers to whom any potential route can be assigned. More in-depth reviews of car-pooling are found in Agatz et al. (2012), Shaheen and Cohen (2019), and Tafreshian et al. (2020).

Car-sharing The term car-sharing refers to a set of services in which customers may book vehicles owned by the car-sharing operator. One may differentiate between station-based and free-floating services. In the station-based case, vehicles are picked up and returned at dedicated stations. This type of car-sharing may be further subdivided into two-way systems, where vehicles must be returned at the same station they were picked up, and one-way systems where pick-up and return may occur at different stations. In free-floating car-sharing, there are no dedicated stations. Instead, vehicles may be picked up and returned anywhere within the area of service. In one-way or free-floating car-sharing systems, one key planning problem is the repositioning of vehicles to match customer demand. These repositioning movements may either be executed by the car-sharing operator or by offering incentives for customers to drop off their vehicles in certain regions. While ostensibly similar to the repositioning problem arising from ride-sharing, there are some differences. Specifically, in car-sharing, there are no dedicated drivers, which adds the additional layer of assigning personnel to reposition vehicles. Moreover, in ride-sharing, vehicles may be shared between multiple customers which increases the complexity when trying to balance supply and demand. These topics are discussed in detail in Chapter 6. Dedicated reviews on car-sharing systems may be found in Jorge and Correia (2013), Ferrero et al. (2018), Huang et al. (2020b), and Golalikhani et al. (2021).

Bike- and scooter-sharing These types of micromobility offerings systems operate in a similar manner as car-sharing. However, due to the involved resources (bikes / e-scooters instead of cars), the associated planning problems and operational processes are different. In general, there are also station-based and free-floating systems. In one-way or free-floating systems, similar repositioning problems as in car-sharing arise. However, due to the possibility of transporting bikes or scooters in bulk with a truck, the operational planning approaches tend to differ. For similar reasons, the proposed algorithms in the context of bike- or scooter-sharing are also not applicable to ride-sharing. For detailed overviews regarding such systems, we refer the reader to Shaheen et al. (2010), Fishman et al. (2013), and Shen et al. (2018).

Ride-sharing The focus of this thesis is on ride-sharing applications. We consider ride-sharing a service in which customers submit a trip request in a similar manner as in taxi or ride-hailing services. However, customers are not transported individually, as their routes may be efficiently combined with other customers with a similar route. Similar to taxi and ride-hailing services, ride-sharing services generally are not station-based although some providers may use designated waiting areas for vehicles or virtual stops for customers. Many service providers such as Uber and Lyft offer both ride-hailing and ride-sharing services. The latter are offered at reduced prices, but the customer must accept the lower level of comfort and increased travel times. A thorough introduction into the field of ride-sharing, different business models and relevant planning problems is provided in the upcoming Chapter 3.

2.3 Summary

In this chapter, we defined the term MaaS as it is used in this thesis. We presented a taxonomy of different mobility services and categorized them according to a set of criteria that influence the design and planning of these services. In particular, we discussed the differences between ride-sharing and other mobility services such as car-sharing or car-pooling. With this basic understanding of mobility-as-a-service, mobility-on-demand, and ride-sharing, we proceed in the following chapter by taking a detailed look at the application setting of ride-sharing and the associated planning problems.

3 Dynamic Ride-Sharing: Application and Planning Problems

In the previous chapter, we have given an overview of different mobility services. In this chapter, we present a more detailed look at dynamic ride-sharing, the focus of this thesis. We present the general application setting and an overview of different business models, stakeholders, and objectives in ride-sharing systems. Finally, we introduce a set of relevant strategical, tactical, and operational planning problems that arise when operating a ride-sharing service.

3.1 Application Setting and Characteristics

We consider ride-sharing to be a service in which multiple groups of customers may be combined into a single trip with multiple customer groups sharing a vehicle simultaneously. This is in contrast to taxi or ride-hailing services where a single customer books a vehicle exclusively for their trip. Additionally, ride-sharing is primarily offered as a profit-oriented business in contrast to car-pooling where the idea is to share private vehicles. Examples of real-world ride-sharing services include UberPool, Lyft Line, GrabShare or MOIA.

3.1.1 Service Characteristics

Ride-sharing services exhibit characteristics that differentiate them from other mobility offerings and influence the business models and planning approaches. In this section, we discuss several key characteristics of ride-sharing services and their impact on the arising planning problems and solution approaches. Furthermore, we specify the type of ride-sharing service that we focus on in this thesis.

Business model While from the customer's point of view the provided services are comparable, the business models of different ride-sharing providers may vary. Depending on the specific business model, a service provider may be faced with different planning problems and operational concerns. Based on the focus of this thesis, the operational planning of ride-sharing services, there are two main criteria to differentiate business models: vehicle ownership and driver employment. On one end of the spectrum are service operators that utilize their own vehicle fleet with employed drivers such as MOIA. These operators maintain direct control over their fleet and drivers. However, they also carry the financial risks associated with vehicle ownership and employee contracts. On the other end of the spectrum are intermediary platforms that mainly act as a broker between independent drivers using their private vehicles and customers (e.g. Uber / Lyft). These broker platforms have limited control over the availability and routes of their drivers and must often work with decentralized incentive mechanisms to influence driver behavior. Other service providers may operate somewhere between the two ends of the spectrum. For instance, it is possible to have employed drivers, but these drivers use their private vehicles to provide ride-sharing services. In this thesis, we focus on business models where the service provider maintains complete control over the vehicle fleet. At times we also discuss decentral control mechanisms, but our main topic is centralized planning algorithms.

High degree of dynamism Ride-sharing services are highly dynamic. In fact, most existing literature in the field considers them as purely dynamic services in which all requests desire immediate service. This is in line with the idea of mobility-on-demand and offers flexibility for customers. At times, it may be desirable to book rides in advance, which is why some service providers such as Uber and Lyft have started to offer reservation options for customers (Uber 2022; Lyft 2022). However, these reservation possibilities are highly constrained and, to our knowledge, only offered for their ride-hailing offerings and not for ride-sharing. In the remainder of this work, we mostly consider purely dynamic ride-sharing. However, in Chapter 5, we also evaluate the impact of pre-booked trip requests.

Trip request demand and response times A general-purpose ride-sharing service must be equipped to handle high levels of trip request demand and still provide near-instantaneous response times for customers. In large cities, such as New York City, a ride-sharing operator may receive more than 20,000 trip requests per hour. Consequently, the operational planning algorithms, in particular for vehicle routing, must be capable of handling these request numbers while still facilitating fast response times.

Limited geographical coverage Ride-sharing is generally offered in limited, urbanized regions. The main reason is that sharing rides is only efficient in regions with sufficient demand. Hence, in the computational evaluations in this thesis, we also focus on urban and suburban areas of operation.

3.1.2 Stakeholders and Obectives

The planning approaches and objectives of a ride-sharing service are strongly influenced by the different stakeholders. We have identified four main stakeholders: service providers, customers, drivers, and cities/regions. In the following, we present their different perspectives on a ride-sharing service.

Service providers In this work, we mainly focus on the perspective of the service provider. The main objective of the service provider is to maximize the profit generated by the ride-sharing service. More often than not, this correlates with other objectives such as the maximization of served trip requests, the minimization of operational costs, and also ensuring customer satisfaction. The latter is important as otherwise customers may resort to alternative modes of transportation.

Customers From the perspective of the customer, there are two main criteria concerning the quality of a ride-sharing service: the price and the service quality. The latter is difficult to measure and encompasses aspects such as waiting and ride times, but also more intangible criteria like the quality of vehicles.

Drivers When considering drivers, one must differentiate between employed and independent drivers. In the latter case, the main objective of drivers is to maximize their personal profit. In contrast, with employed drivers, we would assume a fixed wage, although monetary incentives for serving rides or operating in specific areas are possible.

Cities and Regions More recently, the impact of MaaS offerings of mobility and traffic in cities has received increasing attention. Traffic congestion problems have been attributed to the increasing popularity of ride-hailing services (Castiglione and Cooper 2018) and some cities have begun to take countermeasures by regulating such services (Doubek 2018). These problems can be partially addressed by incentivizing the usage of shared mobility such as ride-sharing. However, the efficiency of mass public transport through metro and buses is still unmatched. Hence, from the perspective of regulators in a city, it is desirable to regulate ride-sharing services in a way that they complement existing public transport solutions, but do not substitute them.

3.2 Planning and Execution of a Dynamic Ride-Sharing Service

In this section, we provide an overview of strategical, tactical, and operational planning problems that arise when operating a dynamic ride-sharing service and review related literature. In addition, we also consider the execution of planned routes and how a service operator may interact with a plan while it is being executed. We discuss potential disruptions and interventions as well as how the gathered data may influence future planning decisions. An overview of planning problems and the route execution is provided in Figure 3.1.

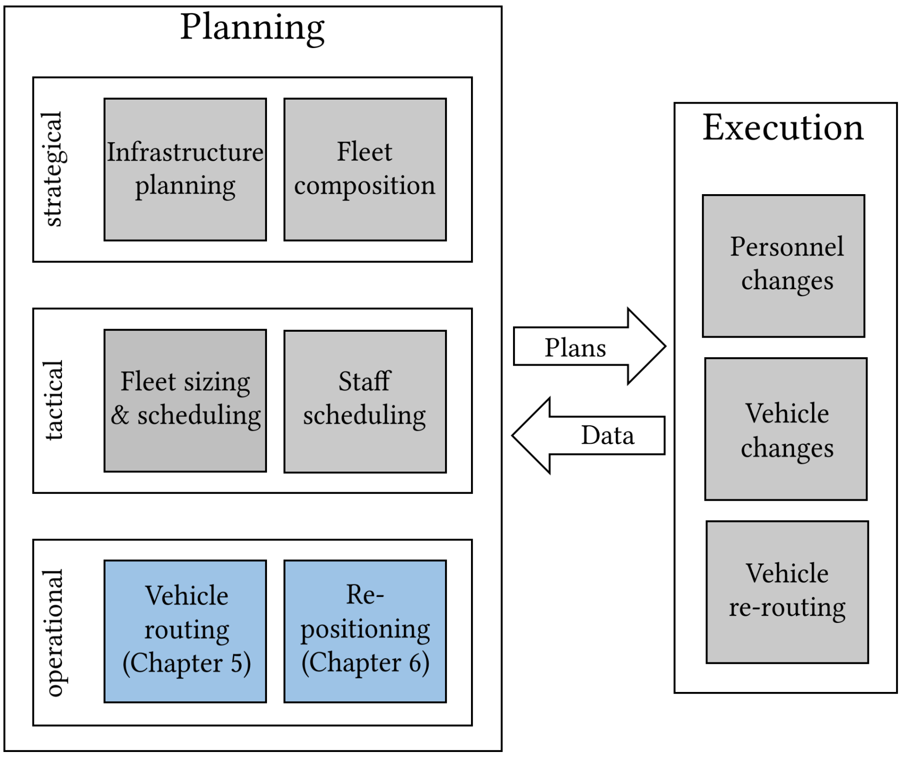


Figure 3.1: Planning problems in dynamic ride-sharing services. The operational planning problems highlighted in blue are considered in detail in this thesis.

3.2.1 Planning Tasks

In hierarchical decision-making, planning tasks are often divided into three categories: strategical, tactical, and operational. Crainic (2000) provides an overview of these planning levels in the context of transportation systems. In the following, we define a set of planning tasks arising in dynamic ride-sharing and categorize them along these three planning levels. On a strategical level, we consider decisions that are made on a long-term basis, often several months or years in advance. The decisions made on this level are typically expensive and cannot be easily modified. Tactical planning is concerned with medium-term decisions and a planning horizon of several days or weeks. Lastly, operational decisions deal with short-term decision-making. In the context of ride-sharing, these are highly dynamic planning problems.

3.2.1.1 Strategical Planning for Ride-Sharing Services

We are not aware of any dedicated works concerning strategical planning for ride-sharing services. We believe that this is mainly due to the fact that strategical problems, such as facility location problems, have been extensively studied in other fields and ride-sharing does not add any novel aspects. Hence, solution approaches from other domains may be transferred. Nevertheless, we shortly discuss relevant strategical planning problems to give a comprehensive overview concerning the planning of ride-sharing services. We consider two planning problems on a strategical level: infrastructure planning and fleet composition.

Infrastructure planning Depending on the business model, it may be necessary for a ride-sharing provider to build and maintain its own infrastructure. In the following, we present related literature regarding three core infrastructure decisions that may arise: the location of vehicle depots, the location of charging infrastructure for electric vehicle fleets, and the location of stops.

The problem of determining depot locations may in its simplest form be modeled as a facility location problem. Reviews on this class of problems are presented in Melo et al. (2009) and Daskin (2013). Alternatively, the depot location problem may be seen as an interdependent problem with vehicle routing. The resulting optimization problem is referred to as the location routing problem (Schneider and Drexl 2017; Drexl and Schneider 2015). To our knowledge, there are no works dealing with the problem of locating vehicle depots in the specific context of dynamic ride-sharing services. However, compared to other vehicle routing variants, the location of depots is of lesser importance when considering ride-sharing. This is mainly due to the fact that there is comparatively little interaction between vehicles and depots. A depot may mainly serve as the start and end of a driver shift and as a parking place for vehicles that are not currently in operation. In contrast, in logistics use cases, vehicles may repeatedly drive to the depot throughout a workday to pick up or deliver goods.

Due to the increasing electrification of vehicles, the location of charging infrastructure for battery-electric vehicles (BEVs) has attracted significant research attention in recent years. For ride-sharing operators, it may be necessary to build and operate their own recharging infrastructure, if it is too costly or infeasible to use public or third-party charging stations. It could be sufficient to provide recharging infrastructure at depots. However, depending on the utilization and range of vehicles that may not be sufficient to provide an adequate service level. There exists a large amount of research on this topic. Extensive reviews are found in Koç et al. (2016), Shareef et al. (2016), and Deb et al. (2018). There are also several works dealing with optimizing the location of charging stations in shared mobility scenarios such as ride-sharing. A comprehensive solution approach and an overview of recent related works is presented in Lokhandwala and Cai (2020).

As a last strategical infrastructure problem, we consider the placement of stops. While this is a highly relevant planning task when operating public transport services, it is of lesser importance for ride-sharing. Ride-sharing services do not operate on fixed lines and while some services use pre-determined stops, these are purely virtual and therefore require no investment into physical infrastructure and may be changed on a relatively short notice. Nevertheless, we see some relevant use cases for the strategical planning of stops in ride-sharing services. Firstly, with ride-sharing services becoming more widespread, regulators may step in and require operators to get approval for stops, which is already the case in Germany with MOIA (MOIA 2022b). Secondly, at some locations (e.g. airports) there already is a requirement for physical stops with separate lanes or parking bays similar to taxi stations. In these cases, the location of stops becomes important for the satisfaction with the provided service, as customers must walk to stops to be picked up by a vehicle. For this problem, one may use existing solution approaches for facility locations (see above) or network design of public transport networks (Desaulniers and Hickman 2007).

Fleet Composition The fleet composition problem, also referred to as fleet design problem, is concerned with determining the composition of a heterogeneous vehicle fleet, i.e. the types of vehicles that are used and the number of vehicles for each type (Etezadi and Beasley 1983). In the context of dynamic ride-sharing, it can make sense to employ different vehicle types for different usage scenarios. For instance, in use cases with dense demand such shuttle services (Lowalekar et al. 2021), one could utilize minibusses for about 10 passengers. On the other hand, in sub-urban areas or during times of lower demand, standard cars with a capacity of 3 − 4 passengers could be more cost-effective as vehicles with a higher capacity would not be fully utilized. As the procurement of a vehicle fleet poses a significant initial investment and vehicles have a considerable running cost, it represents an important strategical decision when running a ride-sharing service. It should be noted that while fleet composition is generally considered as a strategical problem (Baykasoğlu et al. 2019), it may also have a tactical or even operational dimension. For instance, some works consider the inclusion of rental vehicles (Bertoli et al. 2020) which have no associated fixed cost and may be commissioned on a rather short-term basis. While we are not aware of any works dealing with fleet composition specifically for dynamic ride-sharing, there are a number of existing models and algorithms that could be transferred to the use case at hand. For thorough reviews of fleet composition problems and their combination with various classes of vehicle routing problems, we refer the reader to the works of Hoff et al. (2010) and Baykasoğlu et al. (2019).

3.2.1.2 Tactical Planning for Ride-Sharing Services

On a tactical level, we see two types of resources for which tactical plans must be devised: vehicles and drivers. Regarding vehicles, tactical planning is concerned with determining an adequate fleet size throughout each workday and the scheduling of specific vehicles. Closely coupled with these decisions is the scheduling of personnel. Similar to the strategical problems above, we are not aware of any publications discussing these topics specifically for ride-sharing services. However, we believe that solution approaches from other domains may be applied.

Fleet sizing and scheduling The objective of the fleet sizing and scheduling problem is to determine a suitable schedule for the given vehicle fleet such that the number of vehicles in use matches the available demand at any given time. For this purpose, it is necessary to determine which exact vehicles are to be used at which times considering constraints regarding aspects such as vehicle types or regular vehicle maintenance. Such fleet sizing and scheduling problems have been mostly studied in the context of transport logistics (Beaujon and Turnquist 1991) and public transport (Desaulniers and Hickman 2007; Schmid and Ehmke 2015). However, there are also works explicitly considering shared mobility applications (Fagnant and Kockelman 2018). We believe that many of the planning criteria, particularly in the context of public transport, are also relevant in ride-sharing and therefore basic solution principles may be transferred.

Staff Scheduling The staff scheduling problem (sometimes also referred to as rostering) is closely coupled to the fleet sizing and scheduling problem discussed above. Staff scheduling in general is a widely studied problem class with varying applications such as nurse rostering (Burke et al. 2004) or driver scheduling in public transport (Desaulniers and Hickman 2007). For general reviews on staff scheduling, we refer the reader to Ernst et al. (2004) and Van den Bergh et al. (2013). In our opinion, driver scheduling for ride-sharing services does not add any novel aspects compared to existing staff scheduling applications. Hence, we believe that existing solution approaches should be transferable.

3.2.1.3 Operational Planning for Ride-Sharing Services

The focus of this thesis is on operational planning problems for dynamic ride-sharing. In particular, the vehicle routing and repositioning problems. We believe that at the operational level, the special characteristics of the ride-sharing application, such as the high degree of dynamism, come into play and require tailor-made solution approaches.

Vehicle Routing Potentially the most extensively studied problem in the context of dynamic ride-sharing is the arising vehicle routing problem. It is a variation of existing vehicle routing problems and closely related to the dial-a-ride problem (DARP). In this thesis, we refer to it as the vehicle routing problem for dynamic ride-sharing (VRPDRS). While mathematical models and solution approaches for this type of problem have existed for years, recently there has been an influx of papers dealing with the particular challenges arising in dynamic ride-sharing (e.g. Ma et al. (2013) and Alonso-Mora et al. (2017a)). We believe that due to these challenges, particularly the highly dynamic nature and the large number of trip requests, special algorithms for dynamic ride-sharing are necessary and one cannot rely on general-purpose vehicle routing approaches. Hence, in Chapter 5, we present our own solution approach for the VRPDRS and present a detailed literature review.

Repositioning A second important task on the operational level is the management of idle vehicles. The key task is to balance trip request demand and available vehicle supply, i.e. idle vehicles should be repositioned to areas where a lack of available vehicles is anticipated. Similar repositioning problems arise in other mobility applications such as car-sharing (Huang et al. 2020b) or bike-sharing (Reiss and Bogenberger 2017). However, due to the different problem characteristics, we believe that it is beneficial to design specialized solution approaches for repositioning in a ride-sharing setting. The idle vehicle repositioning problem in this context may be either considered in an integrated manner during vehicle routing (Alonso-Mora et al. 2017b) or as a separate optimization problem (Riley et al. 2020). In Chapter 6, we discuss this problem in detail and provide a literature review as well our own solution approach.

3.2.2 Execution of Shared Rides

The plans obtained during operational planning are subsequently executed by drivers and vehicles. In our work, this execution is simulated with a discrete-event simulation framework described in Chapter 4. There is a close interaction between the execution of a plan and the operational planning of vehicle movements. Any updates received concerning the progress of the current plan must be considered regarding future vehicle routes and vehicle movements. Besides directly influencing operational planning, there are also short-term re-planning approaches that may be utilized during the execution of routes to handle disruptions. Disruption handling in general is a widely studied field in several domains such as aviation (Su et al. 2021) or public transport (Lai and Leung 2018). Of particular interest for this work are related works regarding disruptions in vehicle routing (Eglese and Zambirinis 2018). In the following, we discuss potential disruptions of a ride-sharing service and associated disruption handling techniques. It is not within the scope of this thesis to provide an in-depth discussion of disruption handling. We merely aim to give an idea of potential disruptions that may occur and how these can be handled during planning and execution. For a more detailed review of disruption handling for vehicle routing, we refer the reader to Eglese and Zambirinis (2018).

3.2.2.1 Disruption Sources and Types

We differentiate between three sources of disruptions: operator-caused, customer-caused, and external disruptions.

Operator-caused disruptions This category includes any disruptions that are primarily caused by the operator’s resources, in particular drivers and vehicles. This includes, for instance, breakdowns of vehicles or the unplanned absence of personnel.

Customer-caused disruptions Disruptions may also be caused by the customers, for instance, due to no-shows of passengers, changes in the desired destination or short-term cancellations of trip requests.

External disruptions The last category includes any disruptions caused by factors that fall outside the ride-sharing system. These include delays caused by traffic congestion, road accidents, road closures, and service disruptions due to problematic weather conditions.

3.2.2.2 Disruption Handling

In the following, we outline several general options for handling arising disruptions.

Personnel changes In case of short-term disruptions to the driver personnel (e.g. a driver calling in sick at the beginning of the workday), it may become necessary to adjust driver shifts in order to maintain the desired level of service. This may be achieved by re-scheduling the shifts of the own workforce, a problem which has been studied in the context of other applications such as nurse rostering (Wickert et al. 2019). Alternatively, it might be possible to rely on external temporary workers to maintain the service which comes at an extra cost for the ride-sharing provider.

Vehicle changes Vehicle breakdowns are perhaps the most commonly studied disruption type in the context of vehicle routing (Eglese and Zambirinis 2018). Particularly problematic are vehicle breakdowns that occur during the execution of a route. In this case, the provider may opt to supply a backup vehicle that serves the remainder of the disrupted vehicle's route. Such backup vehicles may either be taken from the operator's own vehicle fleet or one can use external rental vehicles. Even if no vehicle breakdowns occur, it may sometimes become necessary to employ additional vehicles on a short-term basis. This may for instance be the case if demand is higher than anticipated.

Vehicle re-routing The most flexible disruption handling technique, as no additional resources are needed, is the re-routing of available vehicles (Eglese and Zambirinis 2018). It can be used to handle most disruption sources, e.g. vehicle breakdowns, no-shows, and delays by adjusting the existing routing plan to consider the arising disruptions. These re-routing decisions can either be taken by a separate module or in the operational vehicle routing algorithm.

3.2.3 Interaction of Planning and Execution

As we have seen in the prior section, there is a close interaction between the execution and planning of a ride-sharing service. Disruptions during the execution may influence the tactical and operational planning levels. In addition, the gathered data during the execution may improve future planning decisions. In the following, we shortly discuss the interaction of execution and planning.

Data-driven planning During the execution, data concerning the routes' progress is sent to the planning services. It is primarily used to supply an up-to-date system state to the operational planning modules. However, one can also use the data in multiple ways to improve future planning decisions. It may, for instance, be used to estimate future travel times and trip request demand. These estimation models can then be used in the planning algorithms to improve decision-making. We present an example of such an integration in Chapter 6, where we utilize a demand forecast to reposition idle vehicles. Besides, obtained data can be used to calculate algorithm parameters on the fly in order to accurately represent the current system state. We employ such an adaptive parameter calculation in Chapter 6.

Disruptions and planning Besides supplying data for future planning decisions, any disruptions that occur during the execution have a direct impact on operational and also tactical planning decisions. On the operational level, the planning modules must be aware of the current state of the route execution to consider this in their decisions. For instance, in case of delays, the vehicle routing algorithm should possibly not assign any further trip requests to a delayed vehicle or even move stops to another route to ensure that no time windows are violated. A disruption of a vehicle or a driver may also trigger re-planning decisions that impact the plan on a tactical level. For instance, if driver shifts are adjusted, this may also impact shifts in the upcoming weeks due to working time considerations.

3.3 Summary

In this chapter, we have introduced the application setting of dynamic ride-sharing and discussed several problem characteristics, stakeholders, and objectives that influence the design of planning algorithms for ride-sharing systems. Subsequently, we presented an overview of the planning and execution of a ride-sharing service. We discussed planning tasks on a strategical, tactical, and operational level and considered the interaction between planning and execution, particularly in the case of disruptions. In the following chapters, we focus on the operational planning level of a ride-sharing service. In the upcoming Chapter 4, we propose a planning and evaluation framework that simulates the execution of a ride-sharing service and enables us to evaluate operational planning algorithms. These planning algorithms themselves are then presented in Chapters 5 and 6.

4 A Modular Planning and Evaluation Framework for Dynamic Ride-Sharing Services

In this chapter, we propose a modular and extensible planning and evaluation framework for dynamic ride-sharing services. This framework is used as a basis for the computational evaluations of our operational planning algorithms for vehicle routing and repositioning in Chapters 5 and 6. At its core, our framework consists of a planning service that maintains the current system state, handles incoming trip requests and plans vehicle movements, and a discrete-event simulation of dynamic ride-sharing (SimDRS) that emulates real-world vehicles, drivers, and customers. In the following, we discuss the design as well as the modular components of our framework in detail. Our focus in this chapter is on the simulation and the communication between the simulation and the planning service. The operational planning algorithms themselves are discussed in detail in Chapters 5 and 6.

4.1 Introduction

When implementing solution algorithms for the operational planning of dynamic ride-sharing services, one generally relies on some form of simulation-based evaluation to assess the performance of the implemented approaches. Unfortunately, the details of these evaluation frameworks are rarely discussed in literature although they may have a large impact on the obtained computational results. Hence, in this chapter, we present a detailed look at the system design of our planning and evaluation framework. We believe that such a simulation-based framework is paramount to operating and planning ride-sharing services. It is essential to evaluate potential solution algorithms for the operational planning of the service. In addition, it can assist service providers in deploying their ride-sharing to a new city or region as it enables them to perform simulation studies beforehand. Therefore, it is our goal to design a simulation and evaluation framework that can be utilized by other researchers and developers in the field of dynamic ride-sharing. In the design of our framework, we pursued the following main goals.

Applicability to large-scale scenarios The simulation should be computationally efficient and enable users to run a variety of large-scale scenarios in an acceptable timespan.

It should support running simulation scenarios faster than real-time in order to conduct simulation studies on large scenario sets.

Modularity and exensibility Components communicate via clearly defined interfaces and can therefore be replaced individually. This modular design enables us to evaluate planning components separately and change the implementation without affecting other components. While we believe that our framework already covers many aspects that are relevant for planning and simulating ride-sharing services, there are still several ways in which it could be improved. Therefore, it should be open to extensions. The external interfaces of our components should also allow users to use several different data sources as a basis to create input data for the simulation.

Decoupling of simulation and planning The simulation and the planning services are decoupled. This would ease the transfer of the implemented planning algorithms to a real-world use case. The simulation is replaced by real vehicles and customers that communicate with the planning service via the same communication interface as the simulation. Additionally, the simulation could be used as a stand-alone solution to evaluate other planning services.

Ease of transfer to new scenarios The data requirements and setup overhead of the simulation should be kept at a minimum to ease the application on new datasets, cities or regions. While the simulation should be able to harness additional data sources, such as travel time information, it should not rely on this data as it may not always be available.

The remainder of this chapter is structured as follows. In Section 4.2, we present an overview of related work concerning the simulation-based evaluation of ride-sharing systems. Section 4.3 presents an overview of our framework and discusses details concerning the planning service and the simulation. Lastly, Section 4.4 summarizes this chapter and presents some possible directions for future extensions.

4.2 Related Simulation Frameworks for Dynamic Ride-Sharing

While most works in the field of dynamic ride-sharing, or more generally dynamic vehicle routing, utilize some form of simulation-based evaluation framework, the details of these systems are rarely discussed. For instance, Ma et al. (2013) and Ma et al. (2015) present a modular system design that is similar to the one proposed in this work. It consists of a planning service and defines communication interfaces with real-world drivers and customers. In their computational evaluations, these drivers and customers are replaced by simulated agents. However, the implementation of this simulation is not presented.

4.2.1 Literature Review

In the following, we discuss several open-source and commercial simulation frameworks that can be utilized to simulate the operation of a ride-sharing fleet and have influenced the design of our own simulation framework. Please note that in this section, we focus purely on the simulation aspect of these works as well as the communication between simulated agents and the service operator. Related approaches for operational planning of ride-sharing services are discussed in Chapters 5 and 6. Furthermore, several approaches discussed throughout this section use traffic simulation models as their basis. While we touch upon some aspects of these models, it is not within the scope of this work to present the basic principles and different variations of traffic simulation in detail. For this purpose, we refer the reader to reviews by Casas et al. (2010) and Wageningen-Kessels et al. (2015). Table 4.1 summarizes related works regarding simulation frameworks for dynamic ride-sharing. We categorize them according to the following criteria.

Traffic flow The traffic flow model determines how vehicles move through the road network. We distinguish between three options. In *microscopic* models, the behavior of each vehicle and interaction between vehicles is modeled on a detailed level and includes aspects such as car-following behavior, lane changes, signals, braking and acceleration. *Mesoscopic* models view traffic flow on a slightly more aggregated level. They still simulate individual vehicles, but the movement on roads is less detailed. These approaches use techniques such as queuing-based or gas-kinetic models in place of vehicle-following models commonly used in microscopic simulations. This leads to decreased computational complexity at the cost of reduced precision. Lastly, in *deterministic* traffic flow models, the interaction between different vehicles on the road is not modeled at all. A vehicle moves according to a deterministic path to its destination. In this approach, one assumes that the relevant aspects of traffic flow, in particular traffic congestion, are encoded in the road network travel times.

Integrated mode choice We differentiate whether or not the simulation offers the option to model choices between different transport modes. This is often possible in approaches that build on more extensive micro- or mesoscopic traffic simulations. The ability to model mode choices is relevant if one wishes to study the interaction between ride-sharing and other modes of transport such as private vehicles, walking, or public transit. Note that while many approaches offer mode choices in theory, this option is not necessarily used in computational studies. It is a time-consuming and data-intensive task to model a transport network with different transport modes. One needs to model the user preferences regarding these modes and integrate additional data sources such as the availability of private vehicles, public transit routes and timetables, etc.

Travel times The travel times on the road network may be either *static* or vary *dynamically* depending on the simulation time and current simulation state. In approaches that use micro- or mesoscopic traffic flow models, travel times are generally dynamic as the time that a vehicle needs to traverse a link depends on the current utilization of this link. In cases where traffic flow is deterministic, one may use time-dependent travel times to achieve dynamism.

Demand Demand for the ride-sharing service may be either given by an *external* data source or generated by the *activities* of the simulated agents. In the former case, one generally uses available data such as records of taxi trips as input for the simulation. In activity-based models, one defines a population of agents and each agent has a set of activities that are carried out over the simulation run. Based on these activities, the transport demands and thereby trip requests for the ride-sharing service are generated.

Customer interaction The interaction between simulated customers and the ride-sharing service may be realized in several ways. Of particular interest is the booking process when a customer submits a new trip request. In the simplest *1-step* variation, the trip request is submitted to the ride-sharing service and the customer is served at some point in the future. No further communication takes place, in particular no confirmation or rejection of the request. These aspects are included in the *2-step* process where the trip request is answered by either a confirmation or a rejection. This gives the ride-sharing provider the option to reject a trip request. In the *3-step* process, the customer additionally has the option to accept or reject the offer made by the ride-sharing provider. This is relevant if multiple options are provided to the customer or the price of the service is not known in advance and instead included dynamically in the offer.

Driver interaction The ride-sharing service also interacts with drivers. In most works, this interaction is merely an *assignment* of vehicle routes to drivers, who subsequently process these routes. One other option is to model *independent* driver behavior. In this case, drivers may have the option to reject a route that has been assigned to them or decide their availability times independently. This is relevant if one wants to simulate applications such as Uber or Lyft in which drivers are not employed by the service provider.

In the following, we discuss the works summarized in Table 4.1. SimMobility (Adnan et al. 2016) is an agent-based simulation framework that consists of three levels. In the long-term and mid-term levels, decisions such as infrastructure location and vehicle ownership are taken and the daily activity and mobility patterns of each agent are built. On the short-term level, SimMobility uses the microscopic traffic simulation MITSIM (Yang and Koutsopoulos 1996) to simulate vehicle behavior with a high resolution. In a study by Azevedo et al. (2016), SimMobility is used to simulate the impact of an autonomous mobility-on-demand service in the inner city of Singapore. The simulation was configured in a fine-grained manner and includes aspects such as public transit routes and timetables, traffic light data and a household survey for activity generation.

POLARIS (Auld et al. 2016) is a mesoscopic traffic simulation framework that uses a similar agent-based activity model as in SimMobility combined with a mesoscopic traffic flow

Name	**References**	**Traffic flow**	**Integrated mode choice**	**Travel times**	**Demand**	**Customer interaction**	**Driver interaction**
SimMobility & MITSIM	Yang and Koutsopoulos (1996) Adnan et al. (2016) Azevedo et al. (2016)	microscopic	yes	dynamic	activity	1-step	assignment
POLARIS	Auld et al. (2016) Gurumurthy et al. (2020)	mesoscopic	yes	dynamic	activity	1-step	assignment
MATSim DRT	Bischoff et al. (2017) Wang et al. (2017) Zwick and Axhausen (2020)	mesoscopic	yes	dynamic	activity	2-step	assignment
MATSim AMoDeus	Ruch et al. (2018) Zwick and Axhausen (2020)	mesoscopic	yes	dynamic	activity	2-step	assignment
MaaSSim	Kucharski and Cats (2022)	deterministic	no	static	external	3-step	independent
SUMO	Qurashi et al. (2020) SUMO (2022b)	microscopic	yes	dynamic	external	1-step	assignment
FleetPy	Kagerbauer et al. (2021) Wilkes et al. (2021) Engelhardt et al. (2022)	deterministic	no	dynamic	external	2-step, 3-step	assignment
SimDRS	Pouls et al. (2020) Pouls et al. (2021) Pouls et al. (2022) this work	deterministic	no	dynamic	external	2-step	assignment

Table 4.1: Related frameworks on simulation for dynamic ride-sharing systems.

model based on kinematic wave theory. Gurumurthy et al. (2020) use POLARIS as a basis for simulating the operation of shared autonomous vehicles in the cities of Bloomington and Chicago. The authors evaluate several detailed simulation scenarios that also integrate other transportation modes such as private vehicles, trains, and buses.

Among the most popular open-source traffic simulation frameworks is MATSim (Horni et al. 2016). It is a mesoscopic traffic simulation that combines an agent-based activity model with a queuing-based traffic flow model. Several works extend MATSim with capabilities to model ride-sharing services. The MATSim DRT and DVRP extensions are used by Bischoff et al. (2017) and Wang et al. (2017) to simulate ride-sharing services in Berlin and Sioux Falls respectively. A similar extension called AMoDeus is presented by Ruch et al. (2018) and used to simulate a scenario in San Francisco based on taxi global positioning system (GPS) trace data. Zwick and Axhausen (2020) compare MATSim DRT and AMoDeus on data from Hamburg provided by the ride-sharing provider MOIA.

Kucharski and Cats (2022) propose a simulation framework called MaaSSim that differs in several aspects from the works discussed so far. Firstly, it has a narrower focus on simulating mobility-as-a-service applications and therefore does not include a micro- or mesoscopic traffic simulation with mode choices and a traffic flow model. Secondly, it is unique among the works presented in this section in that it aims to simulate services like Uber and Lyft with independent drivers. Hence, the authors separately model the behavior of drivers and customers with a service provider platform as an intermediary. Their booking process includes the option for both drivers and customers to reject a ride offer. The authors evaluate MaaSSim on a scenario in the city of Delft.

SUMO (Behrisch et al. 2011) is one of the most popular open-source traffic simulation frameworks besides MATSim. It contains a taxi module (SUMO 2022b) that enables users to model taxi services with the option to activate ride-sharing. At its core, SUMO is a detailed microscopic traffic simulation using a car-following model. Activity-based demand generation is not as tightly integrated as in other traffic simulations discussed in this section. Hence, we consider the demand for ride-sharing as externally given. However, SUMO provides a tool called activitygen (SUMO 2022a) to generate activity-based trips that are used as input for the simulation. Qurashi et al. (2020) use SUMO to simulate an autonomous van-pooling service in the city center of Munich.

FleetPy (Engelhardt et al. 2022) is a recent framework for simulating mobility-on-demand services. Its focus is on realistically modeling the interaction of users and service providers in mobility-on-demand (MOD) services. The authors use their framework to simulate a ride-sharing service in Manhattan. At its core, FleetPy is not a detailed traffic simulation but rather focuses on simulating only MOD services. However, the authors propose combining FleetPy with other tools to facilitate more detailed studies. In Wilkes et al. (2021), they couple FleetPy with mobiTopp (Mallig et al. 2013), a framework for activity-based travel demand modeling. They evaluate their approach on a scenario in Eggenstein-Leopoldshafen. A similar approach is utilized by Kagerbauer et al. (2021), where the authors again use mobiTopp in conjunction with FleetPy. Additionally, they use PTV Visum (PTV Group 2022b) to calculate traffic flows and update travel times on the road network in regular intervals based on the traffic state.

Besides approaches from scientific literature and open-source simulation tools, there are also commercial approaches that offer support for simulating ride-sharing services. Unfortunately, the detailed capabilities and inner workings of these frameworks are often not disclosed. PTV MaaS Modeller (PTV Group 2022a) builds upon the macroscopic traffic simulation PTV Visum (PTV Group 2022b) to enable users to evaluate the impact of shared mobility solutions. A similar solution is provided by Aimsun with Aimsun Ride (Aimsun 2022) which is based on their existing traffic modeling solutions (Casas et al. 2010).

To summarize, there are several existing simulation approaches for dynamic ride-sharing. Several frameworks build upon existing micro- or mesoscopic traffic simulations such as MATSim or SUMO. While these approaches are capable of simulating the operation of a ride-sharing service in great detail, they may pose problems when working with large scenarios due to the involved computational effort. Additionally, these approaches involve a significant overhead when applying them to a new scenario as building a model for a new city or region is data-intensive and time-consuming. There also exist dedicated simulation models for simulating a ride-sharing fleet such as FleetPy. While these offer a reduced level of detail compared to traffic simulations, they are capable of simulating large-scale scenarios and need a minimal amount of data to be transferred to a new application setting.

4.2.2 Contribution

In our simulation framework SimDRS, we opt for a lightweight simulation approach that only simulates the ride-sharing service itself in contrast to a more complex traffic simulation. To the best of our knowledge, it is most similar to FleetPy (Engelhardt et al. 2022) and was developed within the same timeframe. We opt to not include integrated mode choice, complex traffic flow models or activity-based demand generation to reduce the computational complexity of the simulation. In order to facilitate realistic travel times, we enable users to provide time-dependent link travel times for the road network. These may for instance be acquired through data providers such as TomTom (TomTom 2022) or obtained from a traffic simulation. If such data is not available, users may use free-flow travel times obtained from OpenStreetMap (OSM). A similar coupling with traffic simulations as proposed by Kagerbauer et al. (2021) and Wilkes et al. (2021) is also possible in our framework. In this case, the traffic simulation can perform activity-based demand generation and mode choices and communicate these to the ride-sharing simulation. We have performed preliminary experiments with PTV Visum, where an existing Visum model was used to generate different demand scenarios for our simulation runs and additionally provide time-dependent travel times. However, in many cases, no configured traffic simulation model is available. Therefore, these interactions are strictly optional. Concerning customer interaction, we currently implement a 2-step booking process as we assume that all relevant characteristics of the trip such as the price and time constraints can be communicated to the customer before the request is submitted. Hence, no 3-step booking process is necessary. Regarding interaction with the driver, we focus on the use case of ride-sharing with dedicated drivers that cannot independently reject assigned routes. Therefore, in our model, routes and repositioning decisions are assigned to the driver and then carried out as prescribed.

To summarize, we believe that our simulation framework presents a valuable contribution in the field of dynamic ride-sharing due to the following main aspects:

- It is capable of simulating large-scale instances without adding a large overhead due to the running time of the simulation scenarios. This way, developers can evaluate their algorithms under diverse scenarios.
- Our framework is lightweight and does not need large amounts of data. This allows us to quickly set up simulation studies in new cities or regions without needing an extensive data gathering and model configuration phase beforehand.
- Custom planning algorithms for vehicle routing and repositioning are easily integrated, allowing developers to use the framework to evaluate their algorithms. Moreover, if available, they can utilize additional data sources, such as time-dependent travel times, to create more realistic simulation scenarios.

The details of our framework will be presented in the following sections.

4.3 System Overview: Components and Communication

Figure 4.1 presents an overview of the components and data flows in our framework. These components may be partitioned into two separate sections: (1) the planning service, and (2) the simulation SimDRS. The planning service encompasses all modules for the operational planning process of a ride-sharing system. These are described in detail in Section 4.3.1. In addition, it provides communication interfaces for customers and vehicles. The simulation models these two agent types based on various input data sources. This simulation and the different possible data sources are presented in Section 4.3.2. Unless noted otherwise, all software components described throughout the following sections were implemented in C++. However, the focus of this chapter is on the design and structure of our framework and not on the specific implementation details.

4.3.1 Planning Service

The planning service has three core functionalities. Firstly, it serves as a communication interface for customers and vehicles and maintains the current system state based on the messages received from these agents. Secondly, it is responsible for the operational planning of vehicle movements in the form of vehicle routes and repositioning decisions. Lastly, it contains supporting modules to enable these functionalities. In particular, it provides access to a routing engine that calculates shortest paths on the road network.

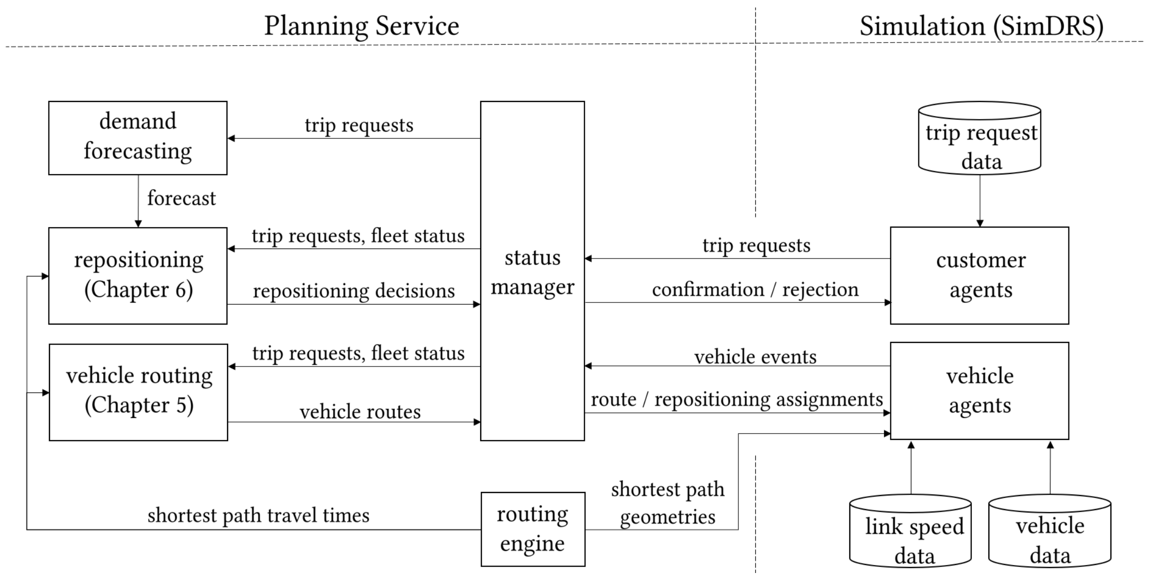


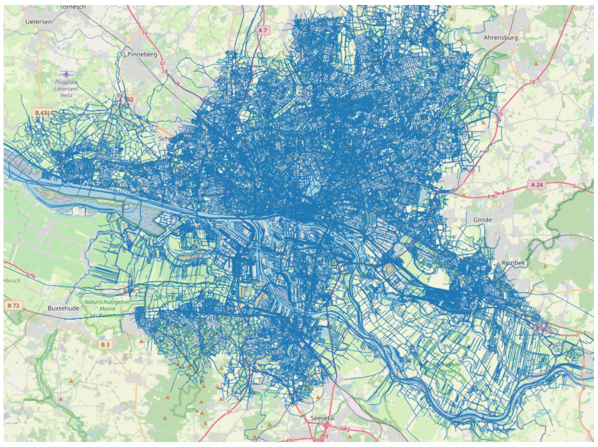
Figure 4.1: Components and data flows of the planning and evaluation framework.

Additionally, it contains a demand forecasting module that provides forecasts for repositioning algorithms. In the following, we discuss the separate modules in the planning service one by one.

Status Manager The status manager handles communication with customers and vehicles. The communication is described in detail in the descriptions of the simulation agents in Section 4.3.2. Based on the information received from the simulation, the status manager maintains the current system state and provides this information to the other planning modules.

Routing Engine The routing engine fulfills two main purposes. Firstly, it provides shortest path travel times between arbitrary locations on the road network to the planning modules. This is, for instance, relevant when checking new insertions into a vehicle route. Secondly, it provides shortest path geometries to the simulation in order to simulate the vehicle movements. The travel times in the simulation are not obtained from the routing engine but rather calculated based on a separate link speed database. This enables the simulation of scenarios with stochastic travel times in which simulated travel times deviate from the travel times used in the planning process. In this work, we utilize RoutingKit (Dibbelt et al. 2016) as our routing engine. This is a fast contraction-hierarchy-based routing solver implemented in C++ and integrated as a library in our project. As a basis for our routing graphs, we use road network data extracted from OpenStreetMap. For this purpose, we define an area under study and extract all roads within this area from OSM data. We select the largest fully connected component in the resulting graph. Exemplary road graphs for three regions used in our computational studies are depicted in Figure 4.2. Currently, we only work with static free-flow travel times in our routing engine. However, it also offers the option of regularly updating the underlying graph based on new travel time information. This would allow us to consider current traffic information during planning. In our studies, this is not evaluated due to the lack of available data.

(a) Chengdu.


(b) Hamburg.

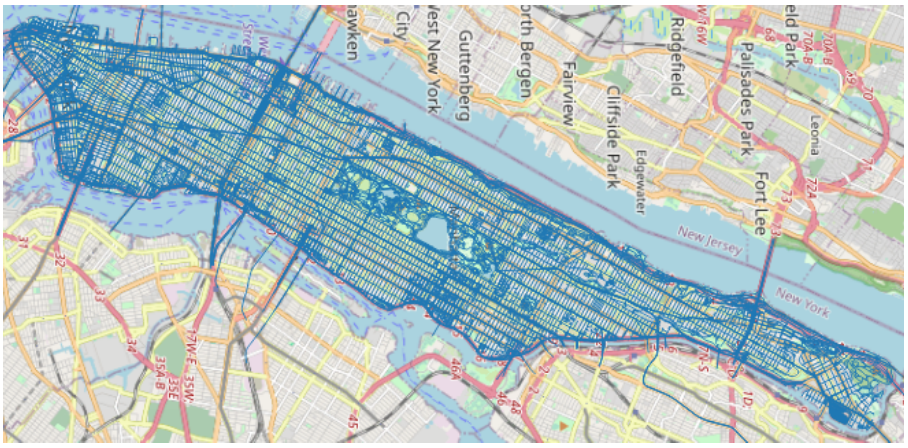

(c) Manhattan.

Figure 4.2: Road networks.

Vehicle Routing The vehicle routing module receives incoming trip requests from the status manager and subsequently plans vehicle routes. The vehicle routing algorithms used in this work are discussed in Chapter 5.

Repositioning and Demand Forecasting The repositioning module repositions idle vehicles to new locations based on a demand forecast. The algorithmic approach, as well as the forecasts used in this work, are discussed in Chapter 6. In this work, we treat repositioning and vehicle routing as separate problems. However, there are also approaches that integrate the anticipation of future requests into the vehicle routing algorithms. In that case, the separate repositioning module would be obsolete and repositioning would be integrated into the vehicle routing module. The advantages and disadvantages of both variations are discussed in Section 6.2.

4.3.2 Simulation

Our simulation is implemented as an agent-based discrete-event simulation that models the behavior of customer and vehicle agents. In the following, we first describe the general simulation procedure and settings. Subsequently, we present the customer and vehicle agents alongside the necessary input data sources.

4.3.2.1 Simulation Process and Settings

The simulation starts by initializing the event queue and agents. This initialization is based on the input data as well as the settings of the specific simulation run. Our simulation framework supports a variety of settings pertaining to trip request parameters and the configuration of the vehicle fleet. We do not discuss all possible settings at this point. Instead, the different simulation scenarios are described in the computational evaluations of Chapters 5 and 6. After initializing the event queue, events are processed one after the other until either no events are left or a defined end of the simulation scenario has been reached. Our framework supports two temporal modes. By default, events are processed as fast as possible. In this case, the next event in the queue is processed immediately after the previous one has finished. This allows for simulation runs that are faster than the simulated real-time equivalent and is useful for computational evaluations. Throughout the remainder of this work, we only use this mode and therefore also focus our descriptions on it. However, we also support a real-time mode. In this case, the simulation may wait until the starting time of the next event in the queue has been reached. This mode can be useful to showcase the behavior of the ride-sharing service in real-time.

4.3.2.2 Customer Agents

The population of customer agents is populated based on external trip request data. This dataset contains the information given in Table 4.2.

Table 4.2: Data structure of trip request data.

Pickup latitude	Pickup longitude	Drop-off latitude	Drop-off longitude	Request time	Passengers
53.557628	9.897047	53.571967	9.7163593	2019-03-16 14:42:00	2
53.462357	9.981423	53.447147	9.9589700	2019-03-19 10:39:00	1
			...		

For each trip in the dataset, a separate customer agent is created. At the given request time, the simulated customer submits the trip request to the planning service. At this point, the request is enriched with the following additional information: a pickup time window, the maximum ride time, and the service time for pickup and drop-off. This enrichment is based on the settings of a specific simulation run and enables users to evaluate a variety of different scenarios. As a response, the customer receives either a confirmation of the request or a rejection in case the trip request cannot be served by the service provider.

4.3.2.3 Vehicle Agents

Vehicle agents are initialized based on an external vehicle dataset that follows the structure given in Table 4.3. It contains the initial location of each vehicle. In our studies, we randomly sample these initial locations from historic trip request data, i.e. the initial vehicle locations correspond to pickup locations of previous trip requests. If available, one could also use other initial locations such as the location of depots. Moreover, the dataset specifies the capacity of each vehicle. Note that we support heterogeneous vehicle fleets. However, the studies in this work are all performed with homogeneous fleets. Finally, the dataset specifies the temporal availability of the vehicle in the form of a start and end time. In theory, this allows for studies with specified schedules for vehicles and a fleet size that varies over time. Please note that in our computational studies, we work with a fixed fleet size, i.e. the complete fleet is available throughout a simulation run.

Table 4.3: Data structure of vehicle data.

Starting latitude	Starting longitude	Capacity	Start time	End time
53.557628	9.897047	4	2019-03-16 08:00:00	2019-03-16 16:00:00
53.462357	9.981423	6	2019-03-16 00:00:00	2019-03-16 08:00:00
			…	

At the specified time, the vehicle registers itself with the planning service. Subsequently, it is available for planning and can receive vehicle routes and repositioning assignments. When receiving a new assignment, the vehicle begins processing it and regularly sends the following progress events to the planning service: regular location events with the current coordinates, arrival at a stop, start of service at a stop, end of service at a stop, departure from a stop, and arrival at a repositioning target. Upon reaching the end of its shift, the vehicle deregisters with the planning service and is no longer available for planning. Progress along the current route is simulated based on two data sources: (1) the shortest path from the vehicle's location to its next target and (2) time-dependent link speeds for this path. This simulation of the vehicle's progress along its route is an essential part of the simulation and therefore described in detail in the following section.

4.3.2.4 Network Data and Travel Times

We assume that the routing engine described in Section 4.3.1 provides shortest path geometries to the simulated vehicle agents. In our studies, we use road network data from OSM to derive these paths. The travel times for each link are obtained from a separate link speed dataset. This dataset provides the information given in Table 4.4. For each link, it contains a set of entries specifying time-dependent speeds on this link. The driving behavior of the vehicle is then simulated based on this speed information. This allows us to simulate dynamic and time-dependent travel times. Note that for the studies in this thesis, we work with static free-flow speeds. In that case, the link speed dataset contains one entry per link with its free-flow speed as obtained from OSM. The main reason for this is the lack of available link speed data for scenarios studied in our computational studies. In practice, such data could be obtained through providers such as TomTom (2022) or via traffic count or speed measurement installations. Alternatively, as mentioned in Section 4.2, a traffic simulation may be used to simulate time-dependent travel times. Overall, this setup would allow users of our framework to simulate scenarios in which the actual travel times during the execution of routes deviate from the planned travel times. This is of particular interest when considering aspects such as disruption handling and reaction to delays.

Table 4.4: Data structure of link speed data.

Arc ID	Start time	End time	Speed
1	2019-03-16 08:00:00	2019-03-16 08:15:00	8.33 $\frac{m}{s}$
1	2019-03-16 08:15:00	2019-03-16 08:30:00	13.89 $\frac{m}{s}$
	...		

4.4 Conclusions

In this chapter, we have presented a simulation-based evaluation framework that enables us to evaluate the performance of dynamic ride-sharing services under a variety of scenarios. We use this framework in the following two chapters to assess the performance of our operational planning algorithms for vehicle routing and repositioning. Our simulation framework covers the main requirements for simulating a dynamic ride-sharing service and includes the option to control relevant aspects such as vehicle availability, trip request settings and road network link speeds in detail.

There are several ways in which the capabilities of the planning service and especially the simulation could be expanded. More detailed modeling of the interaction with both customers and drivers would enable users of our simulation to model application settings with independent drivers and partially decentral decision-making. For this purpose, the booking process could be extended to allow for the rejection of routes by individual drivers. In addition, the communication between the planning service and the simulation should be extended to allow for the option to model a 3-step booking process in which multiple offers are submitted to the customer and the customer may choose one or reject the offer. To enable this, one possible area of future research would be a pricing module in the planning service that determines adequate prices for different ride options. Furthermore, one could extend the simulation with stochastic influences such as cancellations or no-shows. This would allow users to evaluate their planning algorithms under uncertainty. When it comes to travel times, our framework is already capable of handling time-dependent travel times both in the simulation and in the planning process. However, in this work, we do not use this option due to the lack of available data. Hence, it would be promising to build an publish datasets for simulation scenarios that enable the evaluation of time-dependent travel times.

5 Real-Time Vehicle Routing for Dynamic Ride-Sharing

In the previous chapters, we laid the groundwork for the remainder of this thesis by presenting the application setting of dynamic ride-sharing, introducing relevant planning problems, and proposing an evaluation framework that couples planning algorithms with a discrete-event simulation. In this chapter, we focus on one particular planning problem: the vehicle routing problem for dynamic ride-sharing. We present the general requirements, a formal mathematical model and our own solution approach. This chapter is based on the following article:

> M. Pouls et al. (2021). Real-Time Dispatching with Local Search Improvement for Dynamic Ride-Sharing. *Computational Logistics.* Ed. by M. Mes et al. Vol. 13004. Cham: Springer International Publishing, pp. 299–315. DOI: 10.1007/978-3-030-87672-2_20.

5.1 The Vehicle Routing Problem for Dynamic Ride-Sharing

Planning vehicle routes is an essential task when operating a dynamic ride-sharing service. The goal is to assign trip requests to vehicles while meeting a set of planning criteria. From the perspective of the service operator, the main optimization objective is the maximization of profits. This is often approximated by maximizing the number of served trip requests while minimizing the overall operating costs. Commonly, the total vehicle travel time or distance is used as a proxy for these operating costs. On the other hand, from the perspective of the customer, a set of quality of service criteria should be met. Customers expect waiting times, i.e. the time between the submission of a trip request and the arrival of a vehicle, to be within a pre-defined limit. In addition, detours compared to direct travel are acceptable and compensated for by a lower price than traditional taxi services. However, these detours should be kept within specified bounds that may, for instance, be determined relative to the direct travel time. From a mathematical modeling perspective, the VRPDRS may be seen as a dynamic dial-a-ride problem as formulated by Cordeau and Laporte (2007).

Due to the increasing importance of mobility-as-a-service applications and the availability of several datasets from this domain, vehicle routing for dynamic ride-sharing has recently attracted a large amount of research attention (see e.g. Ma et al. (2013), Alonso-Mora et al. (2017a), and Lowalekar et al. (2021)). There are several practical requirements that make the VRPDRS a challenging problem to solve (see also Chapter 3):

- Large problem sizes – Typical datasets for evaluating routing algorithms in the context of dynamic ride-sharing contain over 300,000 trip requests on a single day with peaks of over 20,000 trip requests per hour. Many classical approaches from the field of dynamic vehicle routing are ill-suited for such large instance sizes which makes new solution approaches a necessity.
- High degree of dynamism – Most works from the domain of dynamic vehicle routing assume a moderate degree of dynamism with a portion of trip requests being known in advance. In contrast, the majority of papers dealing with the VRPDRS assume a purely dynamic setting in which all requests arrive dynamically.
- Processing time requirements – Customers of ride-sharing services expect a near-instantaneous response when submitting a trip request. Hence the processing times for trip requests need to be kept small to guarantee a fast response. Some approaches forego this requirement to enable a batching period in which trip requests are gathered. However, this may be problematic from a customer perspective as response times exceeding a few seconds impact the customer's satisfaction negatively.

In this chapter, we present our own solution approach for the VRPDRS. It consists of two separate vehicle routing algorithms that are combined into a real-time planning process. Firstly, we propose a fast dispatching algorithm that processes individual incoming trip requests and facilitates a fast decision regarding whether a trip request is accepted and if so, to which vehicle it is assigned. This dispatching algorithm is combined with a local search improvement phase which is executed repeatedly and strives to improve the routing plan. The main contributions of this chapter may be summarized as follows:

- We introduce a real-time dispatching algorithm based on existing approaches by Ma et al. (2013) and Ma et al. (2015). It processes incoming trip requests and either inserts them into a vehicle's route or rejects them if no feasible insertion is possible. This approach enables us to guarantee fast response times for the customer.
- The dispatching algorithm is used in combination with a local search heuristic that tries to improve the routing plan via simple local search operators. The local search exploits available computational time between the processing of new customer requests.
- Both algorithms are integrated into a planning process that can be used in real-time to facilitate efficient vehicle routing for a dynamic ride-sharing service. We illustrate how this planning process is implemented in our simulation setup, but also show how it could be transferred to a real-world use case.
- We perform an extensive computational evaluation with the simulation framework as presented in Chapter 4. We study the impact of several algorithm parameters and scenario settings such as fleet sizes or customer time windows. Moreover, we also include the possibility of pre-booking trip requests, an aspect that has rarely been

studied in the context of ride-sharing services. Our computational results show that our algorithmic approach can be used in real-time on large datasets and that the utilization of the local search phase improves results compared to purely dynamic dispatching.

In the remainder of this chapter, we will first provide an overview of related literature in the field of dynamic vehicle routing in Section 5.2. Subsequently, in Section 5.3 we present a formal mathematical description of the VRPDRS. Section 5.4 details our vehicle routing algorithms and shows how both algorithms are combined into an overall planning process. Section 7 introduces a simple repositioning algorithm that is used for the computational evaluations in this chapter. Finally, Sections 5.6 and 5.7 present our computational results and the main conclusions of this chapter.

5.2 Related Approaches in Dynamic Vehicle Routing

Dynamic vehicle routing problems have been studied extensively throughout recent years. For general reviews on dynamic vehicle routing problems, we refer the reader to Pillac et al. (2013) and Psaraftis et al. (2016). In this section, we focus mainly on related approaches designed specifically for dynamic ride-sharing systems as these share the closest resemblance to the algorithms proposed in this work. Moreover, we consider related literature in the broader field of dynamic dial-a-ride problems as well as general techniques for local search algorithms in the context of vehicle routing. Note that in this chapter we mainly cover non-stochastic vehicle routing variations without anticipation of future trip requests. This anticipatory component is considered in detail in Chapter 6.

5.2.1 Vehicle Routing for Dynamic Ride-Sharing

In recent years, numerous solution approaches have been proposed for the VRPDRS. These works are most relevant for the problem and solution approach presented in this chapter as they consider the use case of dynamic ride-sharing with purely dynamic trip requests and large instance sizes. As we will see in Section 5.2.2, solution approaches for other vehicle routing variants such as the dynamic dial-a-ride problem (DDARP) consider substantially different application settings. We summarize these works in Table 5.1 based on the following criteria.

Request processing As suggested by Lowalekar et al. (2021), we differentiate between *sequential* and *batch-based* algorithms. Sequential algorithms process trip requests individually in the order they arrive in the system. Batch-based algorithms first gather a batch of requests over a given time period and then process all requests in one batch collectively. This enables them to potentially provide a better solution quality than sequential approaches. However, they are not able to provide an immediate response to the customer.

Planning horizon Most existing approaches are *myopic* in the sense that they only try to find a good solution given the currently known set of trip requests. As this behavior may lead to adverse effects in the future, some works include an *anticipatory (antic.)* component that considers a forecast or some form of stochastic information regarding future trip requests. This can lead to overall improved vehicle routes. Note that the topic of anticipating future trip requests is covered in more detail in Chapter 6.

Objective function The most common objective function is to primarily maximize served trip requests. Often, this is accompanied by the secondary objective of minimizing vehicle travel times or distances. However, this secondary objective is not always explicitly modeled as generally solutions that maximize the number of served trip requests are also efficient concerning travel times. Some works use alternative objective functions such as the minimization of customer waiting times or the maximization of more general utility measures.

Solution approach We provide a short description of the proposed solution methodology. The proposed algorithms are relatively diverse in their nature. However, there are some recurring elements such as the usage of cheapest insertion heuristics or the graph-based solution procedure first proposed by Alonso-Mora et al. (2017a).

Real-world datasets, trip requests and vehicle fleets In addition to this information concerning models and algorithms, we also report some data concerning the computational evaluation of the algorithms. There are several real-world datasets that are used in multiple works, most prominently the New York City taxi dataset (NYC Taxi and Limousine Commission 2022) that we will also use in our evaluations. However, even though multiple papers may draw from the same dataset, the created scenarios tend to differ. Hence, we also report the maximum number of trip requests per time unit. If possible, we give the number of trip requests per hour as this gives us a good impression regarding peak system loads. However, not all papers report this detailed information. In addition to information concerning trip requests, we also report the fleet size and vehicle capacity. While these metrics allow for some form of comparison between the different works and algorithms, the results of the papers vary widely and are not directly comparable. This is mainly due to two reasons. As stated before, some datasets are utilized in multiple works. However, the specific selected time periods, trip requests as well as data filtering and cleaning techniques are different. Hence, the generated scenarios are not identical. More importantly, all works use some form of simulation-based evaluation and the details of these simulations may impact the obtained results in a significant manner. For instance, the assumptions regarding the road network and the assumed travel times are highly relevant. For many papers, this information is not available and hence it is difficult to reproduce or compare results. Some works, such as this one, use free-floating travel times. In our case, these are obtained from OSM data. Other works utilize road networks obtained from OSM, but travel times estimated based on historical trip request data (Santi et al.

2014; Alonso-Mora et al. 2017a). For details on our simulation setup, we refer the reader to Chapter 4.

In the following we discuss the papers summarized in Table 5.1 in detail. Ma et al. (2013) present a real-time system for dynamic ride-sharing. They propose a sequential two-phase insertion heuristic for solving the VRPDRS. Given a new trip request, they first select suitable candidate vehicles based on a spatial-temporal index data structure. This index stores current vehicle positions as well as their predicted future positions according to the current routing plan. In the second phase, they find the cheapest insertion among all candidate vehicles with respect to the additional incurred travel time. In a follow-up work (Ma et al. 2015), the same authors slightly extend their approach by including monetary constraints that ensure that drivers are rewarded for picking up additional passengers. Conversely, prices for passengers are reduced if they have to accept additional detours due to a route insertion. The authors evaluate their approach on instances based on real-world data from Beijing. These contain up to 65,000 trip requests per hour and 7088 vehicles with a capacity of 4. We use this work as a starting point for designing our own vehicle routing algorithm in Section 5.4.

Huang et al. (2014) propose a sequential solution approach based on kinetic trees. Instead of merely storing the route that is currently in execution they also store potential alternative routes for each vehicle. These contain different sequences of the trip requests assigned to a vehicle while satisfying all constraints concerning capacity, time windows, and ride times. For each new trip request their algorithm then finds the cheapest insertion among all potential routes of all vehicles. Thus, the approach may be seen as an extension of the cheapest insertion algorithm utilized in Ma et al. (2013). In order to cope with the resulting computational complexity, the authors design a hotspot-based clustering algorithm that groups stops in spatial proximity to each other. This reduces the size of their tree data structure. The approach is evaluated on real-world data from Shanghai with up to 432,000 trip requests per day and 500 − 20,000 vehicles with a capacity of 3 − 16.

Jung et al. (2016) present a batch-based simulated annealing approach that uses two simple local search operators: moving one customer to another vehicle or swapping two customers between different vehicles. A cheapest insertion algorithm as in Ma et al. (2013) is used to find insertion positions into vehicle routes. The authors evaluate their approach on data from Seoul with up to 4,500 trip requests per hour and 600 vehicles. Note that the instance sizes used in this study are comparatively small and the approach is therefore not necessarily applicable to large-scale scenarios.

A batch-based solution approach is presented by Alonso-Mora et al. (2017a). They collect trip requests over a time period of 30 seconds and subsequently try to insert these requests into vehicle routes with a graph-based algorithm. They employ a graph-based solution procedure. First, they create a pairwise shareability graph of vehicles and trip requests that encodes two types of information: (1) which vehicles can serve which trip requests, and (2) which pairs of trip requests may be combined into a route. Second, based on this information, they create a request-trip-vehicle graph that matches individual trip requests

References	Request processing	Planning horizon	Objective	Solution approach	Real-world datasets	Trip requests	Fleet size	Vehicle capacity
Ma et al. (2013) Ma et al. (2015)	sequential	myopic	max. requests served and min. vehicle travel time	cheapest insertion	Beijing	65,000 / hour	7,088	4
Huang et al. (2014)	sequential	myopic	max. requests served and min. vehicle travel time	branch & bound, MIP, kinetic tree	Shanghai	432,327 / day	500 – 20,000	3 – 16
Jung et al. (2016)	batch	myopic	max. requests served and min. vehicle travel time	cheapest insertion + simulated annealing	Seoul	4,500 / hour	600	4
Alonso-Mora et al. (2017a)	batch	myopic	max. requests served and min. vehicle travel time	graph-based matching	NYC	460,700 / day	1,000 – 3,000	1 – 10
Alonso-Mora et al. (2017b)	batch	antic.	max. requests served and min. vehicle travel time	graph-based matching + request sampling	NYC	460,700 / day	1,000 – 3,000	2 – 4
Cheng et al. (2017)	batch	myopic	max. utility	group-based scheduling	NYC, Chicago	20,000 / hour	100 – 500	2 – 5
Chen et al. (2018)	sequential	myopic	max. requests served and min. vehicle travel time	kinetic tree	Shanghai	432,327 / day	-	4
Tong et al. (2018)	sequential	myopic	max. weighted sum of distance and rejected requests	DP-based cheapest insertion	NYC, Chengdu	517,100 / day	2,000 – 50,000	3 – 20
Lowalekar et al. (2019)	batch	myopic	max. requests served	graph-based matching + zone clustering	NYC	399,695 / day	1,000 – 10,000	1 – 10
Riley et al. (2019)	batch	myopic	min. waiting time	column generation	NYC	32,869 / hour	1,500 – 3,000	1 – 6
Engelhardt et al. (2020)	batch	myopic	max. requests served and min. vehicle travel time	graph-based matching + speed-up	Munich	180,000 / day	200 - 3,000	4
Shah et al. (2020)	batch	antic.	max. requests served	graph-based matching + RL & ADP	NYC	19,820 / hour	1,000 – 3,000	2 – 10
Lowalekar et al. (2021)	batch	antic.	max. requests served	graph-based matching + zone clustering + benders decomposition	NYC	403,770 / day	1,000 – 10,000	1 – 10
Pouls et al. (2021) this work	sequential	myopic	max. requests served and min. vehicle travel time	cheapest insertion + local search	NYC, Hamburg, Chengdu	429,855 / day	72 – 1,512	2 – 6

Table 5.1: Related literature on vehicle routing for dynamic ride-sharing.

and vehicles to routes, i.e. combinations of trip requests. On this graph, they solve a matching problem that matches individual trip requests and vehicles to routes with the objective of maximizing the number of served trips and minimizing the necessary travel times. The general structure of this graph-based approach is used in several other works (Alonso-Mora et al. 2017b; Lowalekar et al. 2019; Shah et al. 2020; Lowalekar et al. 2021). In their own follow-up paper (Alonso-Mora et al. 2017b), the authors propose an anticipatory variation that includes sampled anticipated trip requests. These requests are served with a lower priority than actual trip requests. Both approaches are evaluated on real-world data from Manhattan with up to 460,700 trip requests on a single day and a fleet size between 1,000 and 3,000 vehicles with a capacity of $1-10$.

Cheng et al. (2017) propose an alternative objective function for the VRPDRS that aims to incorporate the satisfaction of drivers and customers. The authors introduce a utility measure that considers the preference of customers for vehicles, other passengers that they share a vehicle with and the planned trajectory. The authors present several algorithms to solve the VRPDRS with this objective function. The most successful one is a batch-based approach that first groups similar trip requests and subsequently schedules them collectively. A computational evaluation is performed on a synthetic dataset as well as instances derived from real-world data from NYC and Chicago. These contain up to 20,000 trip requests per hour and $100-500$ vehicles with a capacity between 2 and 5.

Chen et al. (2018) present an algorithm that integrates elements from the cheapest insertion approach by Ma et al. (2013) and the kinetic tree algorithm by Huang et al. (2014). The key novelty in their approach is that it finds a set of potential vehicles for each incoming trip request that are non-dominated concerning the price or pick-up time. This gives the customer the option to choose between multiple alternatives. They evaluate their approach on data from Shanghai with up to 432,327 trip requests within 24 hours.

Tong et al. (2018) present another insertion-based solution approach for the VRPDRS. Their work contains two key innovations. Firstly, they consider a generalized objective function that consists of a weighted sum of travel distance and rejected trip requests. Secondly, they propose a linear time insertion feasibility check based on a dynamic programming approach. They evaluate their algorithm on instances derived from real-world data from NYC and Chengdu with 2,000 - 50,000 vehicles and a capacity of 3 - 20. The number of trip requests per day is up to 517,100 for the NYC dataset and 259,347 for Chengdu.

Lowalekar et al. (2019) introduce a batch-based solution approach that uses the algorithm by Alonso-Mora et al. (2017a) as a foundation. Instead of matching vehicles to combinations of requests, they first cluster locations of pickup and dropoff locations into zones. Subsequently, they determine paths through zones and then match vehicles to these paths. An extended non-myopic variation of this algorithm is proposed by Lowalekar et al. (2021). Here the authors design a benders decomposition approach that incorporates information about anticipated trip requests while assigning vehicles to zone paths. They evaluate their approaches on real-world data from NYC as well as another real-world dataset from an undisclosed location and an artificial dataset. These scenarios contain up to 403,770 requests per day and are evaluated with $1{,}000-10{,}000$ vehicles with a capacity of $1-10$.

A column generation batch-based algorithm is presented by Riley et al. (2019). One key difference compared to our work and all other papers discussed in this section is that they do not permit the rejection of trip requests. Consequently, their objective function is to minimize the waiting times of customers while serving all incoming requests. In order to cope with the real-time requirements of the application setting, they utilize an anytime pricing algorithm that is guaranteed to return a feasible solution even if interrupted due to running time limits. They evaluate their algorithm on NYC data with up to 32,869 trip requests per hour and 1,500 – 3,000 vehicles with a capacity between 1 and 6.

Engelhardt et al. (2020) introduce an extension of the graph-based solution approach by Alonso-Mora et al. (2017a). The key innovation is to maintain information concerning the feasibility of potential vehicle routes between the processing of batches of new trip requests. This way, the computational efficiency of the algorithm can be improved. The authors evaluate their work on a generated dataset in the city of Munich with up to 180,000 trip requests per day and 3,000 vehicles with a capacity of 4. The authors show that their algorithm outperforms a simple insertion heuristic and their speed-up approaches can reduce the computational time for some algorithm phases by a large amount which can be useful during peak demand times.

Shah et al. (2020) propose another variation of the graph-based approach by Alonso-Mora et al. (2017a). They again build upon the same basic graph-based algorithm. However, they combine reinforcement learning and approximate dynamic programming in order to score potential routes and update their score function. They then assign vehicles to routes in a way that maximizes these scores. The score of a route includes an approximation of the future value of the decision. Hence, the approach is anticipatory and able to consider the presumed impact of a decision. The authors perform a computational evaluation on data from NYC with up to 19,820 requests per hour and 1,000 – 3,000 vehicles with a capacity between 2 and 10.

5.2.2 Dynamic Dial-a-Ride Problem

From a mathematical modeling perspective, the VRPDRS may be seen as a dynamic dial-a-ride problem (DDARP). There exists a large amount of prior research regarding this problem class. For extensive literature reviews specifically regarding dial-a-ride problems (DARPs), we refer the reader to Cordeau and Laporte (2007) and Molenbruch et al. (2017). As the DARP itself is a variant of the pickup and delivery problem, it is also covered in several more general reviews concerning dynamic vehicle routing (Parragh et al. 2008; Berbeglia et al. 2010; Pillac et al. 2013; Psaraftis et al. 2016; Ritzinger et al. 2016). However, while the underlying problem model may be similar, most traditional works on the DDARP focus on applications other than dynamic ride-sharing. Classical applications include, for instance, the transportation of patients in hospitals (Beaudry et al. 2010) or mobility services for the elderly and people with disabilities (Borndörfer et al. 1999). The requirements of these application settings differ substantially from the use case of dynamic ride-sharing that we described in Chapter 3.

Firstly, considered problem sizes as given by the number of requests over time and the size of the vehicle fleet are considerably smaller. Frequently used benchmark datasets (Cordeau and Laporte 2003; Cordeau 2006) contain instances with 16 - 144 customers. A survey by Parragh et al. (2008) found that the largest instances used to evaluate DDARP algorithms contained merely around 4,000 customers within 24 hours. In contrast, in the setting of dynamic ride-sharing, we often consider around 300,000 trip requests in the same time period. Secondly, realistic VRPDRS scenarios have some characteristics that are not prevalent in many other applications. In the dynamic ride-sharing setting, we consider a highly dynamic use case. Many studies in this area even focus on purely dynamic scenarios where all requests arrive dynamically. In contrast, many DDARP publications consider a varying degree of dynamism where only a fraction of requests arrives dynamically. Moreover, in dynamic ride-sharing, one generally assumes relatively tight constraints on the pickup time windows and the maximum ride time of customers. Due to these differences between classical DDARP applications and the dynamic ride-sharing setting, most solution approaches for dynamic dial-a-ride problems are not directly applicable to the dynamic ride-sharing use case.

5.2.3 Local Search Approaches for Dynamic Vehicle Routing

An extensive amount of prior research exists on local search algorithms for vehicle routing problems. As noted in the review by Funke et al. (2005), many heuristics and metaheuristics for vehicle routing problems (VRPs) rely on some form of local search. Generally, these algorithms consist of two main components: (1) local search operators (also referred to as neighborhoods) and (2) a search algorithm. The local search operators are steps that modify the solution in some way in an attempt to achieve a better solution while still meeting all planning criteria and constraints. For instance, moving a stop to a different position in the same route or removing the stops associated with a customer from a route and re-inserting them into another route. The search operators may range from such simple moves to significantly more complex move chains in which several routes are modified. In this work, we employ relatively simple search operators due to the running time constraints present in the dynamic ride-sharing setting. The second component of a local search approach is a search algorithm that coordinates and steers the search. In its most simplistic form, this may be a neighborhood descent heuristic that iteratively applies search operators until no further improvements are found. More complex popular search algorithms are, for instance, tabu search (Glover and Laguna 1998), variable neighborhood search (Hansen and Mladenović 2003) or simulated annealing (Bertsimas and Tsitsiklis 1993). In this work, we apply a rather simple search scheme that combines an iterative neighborhood descent with a tabu mechanism to reduce running times and diversify the search. For more thorough overviews concerning local search operators and algorithms in the context of vehicle routing, we refer the reader to Funke et al. (2005), Cordeau et al. (2008), and Groër et al. (2010). Note that most of the mentioned works cover local search approaches in a static environment. When utilizing such approaches in a highly dynamic environment, some challenges arise regarding the integration into the dynamic planning process. These include the interruption of the search due to newly arriving events and the transfer of improved solutions to vehicles. Some works (Attanasio et al. 2004; Beaudry et al. 2010) address these issues. However, they are working with application settings in which the frequency of incoming trip requests is significantly lower than in the ride-sharing setting.

5.2.4 Contribution

In the previous sections, we have presented an overview of solution approaches for dynamic vehicle routing problems with a focus on the VRPDRS. Compared to the current state-of-the-art, we introduce several novel aspects in our work that have not been extensively studied yet. We believe that our algorithmic approach is well-suited to handle the challenges arising from the ride-sharing setting that we introduced in Chapter 3. Our main contributions may be summarized as follows.

Combining sequential dispatching and local search improvement To our knowledge, all existing algorithms for the VRPDRS are either purely sequential or batch-based. While batch-based approaches have the potential of providing a superior solution quality, they have the inherent disadvantage that the customer does not receive an immediate response to their trip request. Sequential approaches on the other hand may take sub-optimal decisions that cannot be reverted due to the structure of the algorithm. In this thesis, we combine a sequential dispatching algorithm that accepts or rejects trip requests with a local search improvement procedure that utilizes available computational time to improve the current routing plan.

Integration of a local search into a real-time planning process As mentioned previously, there are few works dealing with the integration of local search techniques into a real-time planning process for large-scale dynamic vehicle routing problems. We believe that it is promising to utilize widely studied local search approaches in such a dynamic setting. Hence, we propose the integration of a simple local search into our dynamic planning setting and also illustrate the future potential of integrating more complex metaheuristics.

Evaluation and pre-bookings We evaluate our approach on multiple real-world datasets and study several practically relevant aspects such as the impact of time windows and fleet sizes. We introduce aspects in our evaluation that have been rarely studied in the context of ride-sharing services. For instance, we investigate the impact of trip requests that are reserved in advance. While most algorithmic approaches are theoretically capable of processing such requests, we are not aware of any structured evaluation how such pre-booked trips influence the solution quality.

5.3 The Vehicle Routing Problem for Dynamic Ride-Sharing

The vehicle routing problem in a dynamic ride-sharing setting may be formulated as a dynamic dial-a-ride problem. In this section, we present a formal mathematical formulation of the problem based on the static DARP formulation by Cordeau and Laporte (2007). We first introduce the necessary notation (summarized in Table 5.2) and subsequently formulate a mixed-integer programming model. While we do not actually solve this mathematical formulation, we believe that it is useful for gaining a formal understanding of the problem. The notation introduced in this section is also utilized throughout the algorithm description in Section 5.4.

Let R be the set of all trip requests. Each entry $r \in R$ denotes an individual trip request for a given number of passengers q_r as well as a pickup and drop-off location p_r and d_r respectively. Furthermore, it has a service time s_r that occurs at the pickup and drop-off service. Each request has a creation time t_r at which it enters the system. Moreover, it has a pickup time window specified by the earliest pickup time $e_{p,r}$ and the latest pickup time $l_{p,r}$. The size of the time window is referred to as the maximum waiting time w_r of a customer. We assume that most customers desire immediate service, i.e. $e_{p,r} = t_r$. However, we also allow for the option of pre-booking requests with an earliest pickup time $e_{p,r}$ at some point in the future. We then refer to the time interval between t_r and $e_{p,r}$ as the pre-booking time pb_r. In addition to the time window on the pickup of a customer, there is also a temporal constraint on the customer's ride time given by the maximum ride time L_r. This is motivated by the fact that, although ride-sharing allows for detours compared to direct taxi services, these detours should be limited. In practice, the acceptable detour could either be specified by the customer or determined by the operator. In this work, the maximum ride time is determined as $L_r = \max(m^{det} \cdot tt_{p_r,d_r}, tt_{p_r,d_r} + L^{min})$. Here, tt_{p_r,d_r} denotes the direct travel time between pickup and drop-off of the request while $m^{det} > 1.0$ corresponds to a detour factor that allows for a relative detour compared to this direct travel time. For trip requests covering only very short distances, we introduce a minimum additional ride time L^{min}. Otherwise, these trip requests are very difficult to combine with other requests. While the maximum ride time is the constraining factor on the timing of the drop-off stop, we sometimes use an explicit time window that does not depend on the actual pickup time. Hence, we also determine the earliest possible drop-off as $e_{d,r} = e_{p,r} + s_r + tt_{p_r,d_r}$ and the latest possible drop-off $l_{d,r} = l_{p,r} + s_r + L_r$. To serve the trip requests R, we employ a fleet of vehicles K. Each vehicle $k \in K$ is associated with a capacity Q_k and a starting location v_k^0.

As we are dealing with the dynamic version of the problem, the trip requests R may be partitioned into three distinct subsets $R = R^n \cup R^a \cup R^p$. R^n denotes newly arrived trip requests. If each individual trip request is processed at the exact moment it enters the system, this set consists of precisely one trip request per decision epoch. However, it would also be imaginable to have a batching period in which requests are collected and subsequently processed together. R^a contains those trip requests which were previously accepted and assigned to vehicles, but are still awaiting their pickup. Lastly, R^p consists of

Table 5.2: Notation for the VRPDRS.

Sets	
K	Vehicles
V	Nodes
P/D	Pickup / drop-off nodes
R	Requests
$R^n/R^a/R^p$	New / assigned / picked up requests
Parameters	
s_i	Service time at node i
$tt_{i,j}$	Travel time between nodes i and j
tt^{max}	Maximum travel time between any pair of nodes
W^{req}	Objective function weight for serving requests
q_i	Load change at node j
$e_{p,r}/e_{d,r}$	Earliest start of pickup / start of drop-off of r
$l_{p,r}/l_{d,r}$	Latest start of pickup / start of drop-off of r
t_r	Creation time of r
pb_r	Pre-booking time of r
w_r	Maximum waiting time of r
L_r	Maximum ride time of request r
L^{min}	Minimum additional detour
m^{det}	Detour factor
Q_k	Capacity of vehicle k
p_r/d_r	Pickup / drop-off node of r
k_r	Vehicle k by which a request $r \in R^p$ is served
v_k^0	Initial location of vehicle k
ps_r	Actual start of pickup service for $r \in R^p$
Decision variables	
$x_{i,j}^k$	1, iff vehicle k traverses the arc (i, j)
ar_i^k	Arrival time of vehicle k at node i
de_i^k	Departure time of vehicle k at node i
w_i^k	Capacity utilization of vehicle k after node i
rt_r^k	Ride time of request r in vehicle k

all requests that have already been picked up by a vehicle. Trip requests $r \in R^a \cup R^p$ are associated with their assigned vehicle k_r. In addition, requests $r \in R^p$ are also associated with the actual start of their pickup service ps_r as it has already occurred.

Given these elements, we can formulate the VRPDRS on a complete directed graph with nodes V corresponding to the pickup ($P = \bigcup_{r\in R} p_r$) and drop-off ($D = \bigcup_{r\in R} d_r$) locations as well as the initial vehicle locations v_k^0 for $k \in K$ and a dummy node v_∞. This dummy node is used as the final destination of all vehicles and is only needed for modeling the problem. Each node $i \in V$ in the graph is associated with a service time s_i and passenger change q_i. For the pickup and drop-off nodes these correspond to the service time of the corresponding request s_r and the number of passengers boarding or alighting the vehicle respectively. For the nodes corresponding to the initial vehicle locations, a distinction has to be made between different cases. Either a vehicle $k \in K$ is en route to its next stop, in which case the service time $s_{v_k^0}$ and passenger change $q_{v_k^0}$ at its initial location are zero. The same is true for idle vehicles. However, if a vehicle is currently serving a request at a stop, the service time is set to the remaining service time and the load change corresponds to the passenger change at the stop. Arcs in the graph (i, j) for $i, j \in V$ have an associated travel time $tt_{i,j}$. Arcs involving the dummy node v_∞ have a travel time of $tt_{i,v_\infty} = 0$ for $i \in V \setminus v_\infty$.

The mathematical model given in Equations 5.1 −5.17 contains five sets of decision variables. The main set of binary decision variables assigns vehicles to arcs in the graph and is defined as follows:

$$x_{i,j}^k = \begin{cases} 1 & \text{if vehicle } k \in K \text{ traverses arc } (i,j) \text{ where } i,j \in V \\ 0 & \text{else} \end{cases}$$

Moreover, there are four sets of auxiliary variables which are used to ensure bounds on the time windows, vehicle loads and ride times:

ar_i^k, guaranteed to be ≥ the arrival time of vehicle $k \in k$ at node $i \in V$

de_i^k, guaranteed to be ≥ the departure time of vehicle $k \in k$ at node $i \in V$

w_i^k, guaranteed to be ≥ the passenger occupation of vehicle $k \in K$ after node $i \in V$

rt_r^k, guaranteed to be ≥ the ride time of request $r \in R$ in vehicle $k \in K$

In contrast to classical static DARP formulations (Cordeau and Laporte 2007), we allow for the rejection of trip requests, if these cannot be feasibly inserted into the current vehicle routes. Therefore, we use a hierarchical objective function 5.1 which primarily maximizes the number of served requests and secondarily minimizes the total driving time. The prioritization of the first objective is ensured by choosing an adequate objective weight of $W^{req} = 4 \cdot tt^{max} + 1$ where tt^{max} corresponds to the maximum travel time between two nodes. As accepting an additional request will add at most four traversed arcs to the solution, the incurred additional travel time is capped by $4 \cdot tt^{max}$. Thus, our choice of weights ensures that accepting an additional trip request is always beneficial for the overall objective function value. Constraints 5.2 ensure that nodes belonging to new requests are visited at most once, as these may be rejected if no suitable vehicle is available. On the other hand, Constraints 5.3 and 5.4 guarantee that assigned customers, as well as picked-up customers, must be visited. In the latter case, the vehicle by which the service is provided is already fixed, whereas in the case of merely assigned requests we still allow for the possibility of the requests being moved to another vehicle. Departure from the initial vehicle position is enforced by Constraints 5.5, while Constraints 5.6 ensure that all vehicles arrive at the dummy node as their final position. Constraints 5.7 guarantee that

$$
\begin{array}{llll}
\text{(VRPDRS)} & W^{req} \sum_{r\in R^n} \sum_{k\in K} \sum_{i\in V} x^k_{i,p_r} - \sum_{k\in K} \sum_{i\in V} \sum_{j\in V} tt_{i,j} x^k_{i,j} & \rightarrow \max & (5.1)\\
\text{s.t.} & \sum_{k\in K} \sum_{i\in V} x^k_{i,p_r} \leq 1 & r \in R^n & (5.2)\\
& \sum_{k\in K} \sum_{i\in V} x^k_{i,p_r} = 1 & r \in R^a & (5.3)\\
& \sum_{i\in V} x^{k_r}_{i,d_r} = 1 & r \in R^p & (5.4)\\
& \sum_{j\in V} x^k_{v^0_k,j} = 1 & k \in K & (5.5)\\
& \sum_{i\in V} x^k_{i,v_\infty} = 1 & k \in K & (5.6)\\
& \sum_{i\in V} x^k_{i,p_r} - \sum_{i\in V} x^k_{i,d_r} = 0 & r \in R^n \cup R^a, k \in K & (5.7)\\
& \sum_{j\in V} x^k_{j,i} - \sum_{j\in V} x^k_{i,j} = 0 & i \in P \cup D, k \in K & (5.8)\\
& de^k_i \geq ar^k_i + s_i & i \in V, k \in K & (5.9)\\
& ar^k_j \geq (de^k_i + tt_{i,j}) x^k_{i,j} & i,j \in V, k \in K & (5.10)\\
& w^k_j \geq (w^k_i + q_j) x^k_{i,j} & i,j \in V, k \in K & (5.11)\\
& rt^k_r \geq ar^k_{d_r} - de^k_{p_r} & r \in R^n \cup R^a, k \in K & (5.12)\\
& rt^k_r \geq ar^{k_r}_{d_r} - (ps_r + s_r) & r \in R^p & (5.13)\\
& e_{p,r} \leq ar^k_i \leq l_{p,r} & r \in R^n \cup R^a, k \in K & (5.14)\\
& rt^k_r \leq L_r & r \in R, k \in K & (5.15)\\
& w^k_i \leq Q_k & i \in V, k \in K & (5.16)\\
& x^k_{i,j} \in \{0,1\} & i,j \in V, k \in K & (5.17)
\end{array}
$$

service at the pickup and drop-off nodes of a request is provided by the same vehicle. Flow conservation at pickup and drop-off nodes is enforced by Constraints 5.8. Constraints 5.10 – 5.13 ensure that the decision variables for arrival times, load and ride times are chosen consistently. Variables denoting the arrival time at nodes are set appropriately by Constraints 5.10. We ensure that the arrival time of the vehicle $k \in K$ serving node $j \in V$ is bounded by the arrival time at the previous node $i \in V$ plus the service time at i and the travel time from i to j. Similarly, Constraints 5.11 set the load of vehicle k after visiting node j to be larger or equal to the load after the prior node i plus the load change at node j. The same procedure is performed for the ride time variables in Constraints 5.12 and 5.13. In the case of new and assigned requests, the ride time rt^k_r is constrained by the difference between the planned arrival times at the drop-off and pickup nodes minus the service time at the pickup node. Special treatment is provided for requests that were already picked up $r \in R_p$ in Constraints 5.13. In this case, the planned arrival time at the pickup node is replaced by the actual arrival at the pickup node ps_r. The limits on the pickup time windows, ride times and vehicle capacities are defined in Constraints 5.14 – 5.16. The arrival time at a pickup node p_r must lie within the defined time window $[e_{p,r}, l_{p,r}]$ as given in Constraints 5.14. In a similar fashion, the ride time is capped by the maximum allowed ride time of a given request in Constraints 5.15. Constraints 5.16 ensure that the vehicle capacity is never exceeded. Finally, the variable domain for binary decision variables $x_{i,j}^k$ is given by Constraints 5.17. With this mathematical model, we have presented a formal definition of the VRPDRS. In the next section, we follow this up with our solution algorithm for the presented problem.

5.4 Real-Time Dispatching with Local Search Improvement

Our solution approach for the VRPDRS consists of two algorithms:

1. A real-time dispatching algorithm (DIS) that is responsible for immediately handling incoming trip requests and assigning them to vehicles if possible. This algorithm decides on the acceptance or rejection of a trip request and enables us to provide fast response times for the customer.
2. A local search improvement algorithm (LS) that tries to improve the current routing plan through local search operators. The local search exploits available computational time whenever no trip request is being processed.

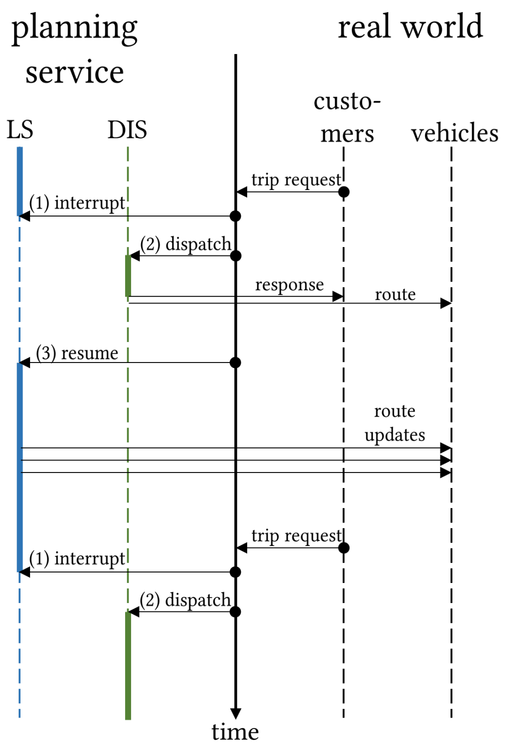
In the following, we first present a detailed look at our planning process, both in the envisioned practical implementation and in a slightly modified variant for simulation studies. Subsequently, we describe our dispatching and local search algorithms.

5.4.1 Planning Process

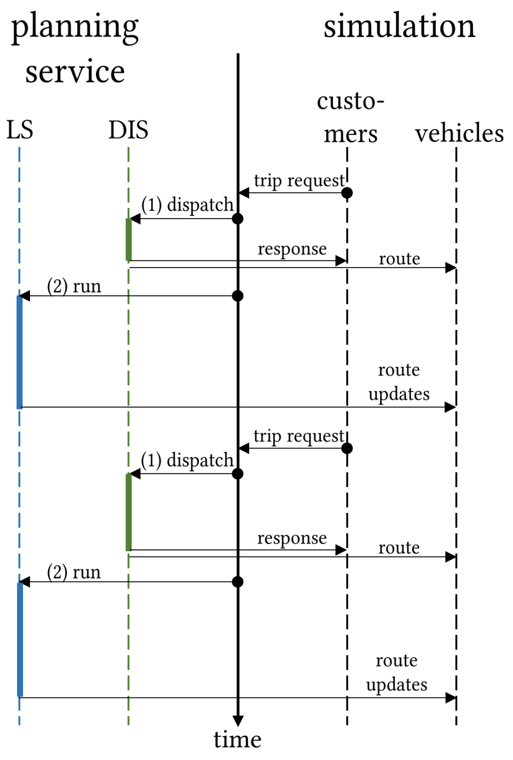
Our planning process is illustrated in Figure 5.1. There are two variations: (1) the envisioned usage in a real-world ride-sharing service, and (2) a slightly adapted version for simulation studies.

5.4.1.1 Real-World Usage

Whenever a new trip request is submitted by a customer, the first step is to interrupt the continuously running local search. Subsequently, the trip request is processed by the dispatching algorithm. As a result, the algorithm returns a response to the customer whether or not the request can be served. In case of an accepted trip request, it additionally



(a) Real-world setup.



(b) Simulation setup.

Figure 5.1: Planning process in the real-world setup and the simulation environment.

returns an altered vehicle route into which the new trip request has been inserted. Note that DIS typically terminates after less than 10 ms, thereby providing a nearly instant response to the customer. After handling a request, the local search is resumed until the next trip request arrives in the system. Any improvements made by the local search are immediately sent to the affected parties, i.e. vehicles and customers.

5.4.1.2 Simulation Usage

For our simulation-based evaluations, we perform simulation runs faster than real-time in order to allow for the evaluation of a large set of diverse scenarios. Hence, in the basic setup, each trip request is processed immediately after the previous one has finished processing. As this would leave no time for the local search algorithm, we adapt the planning process slightly. After dispatching a trip request, we run LS with a fixed time limit. Subsequently, the next trip request is processed. An adequate time limit may be derived from the peak number of trip requests per hour for any given scenario. The time limit should be set low enough that the simulated time still runs faster than real-time.

5.4.1.3 Design Decisions

In the combination of dispatching and local search as outlined above, we made two key design decisions that we would like to explain shortly. Firstly, we opted to not introduce a batching period and process multiple trip requests at once. As already outlined in Section 5.2, this was done to ensure an immediate response for customers. Additionally, in our setup, the local search phase is responsible for optimizing the routing plan as a whole. Hence, it is not necessary for the dispatching algorithm to change the assignment of multiple trip requests at once. Secondly, we elected to run DIS and LS sequentially and not in parallel. Theoretically, one could run the local search continuously parallel to the dispatching algorithm. However, in practice this would lead to complications as the local search could modify the current routing plan at the same time as the dispatching algorithm inserts a new trip request. This could invalidate the routing plan if both modifications are incompatible. To resolve this conflict, we would either need intricate communication between the two algorithms to keep the routing plan in sync. Alternatively, there could be a periodic merging process that merges the changes made by the local search into the plan that is currently in execution. Given these restrictions, we see no advantage in running both algorithms in parallel. In our sequential setup, the interruptions of the local search are short due to the low running time of the dispatching algorithm. Hence, we only lose a minor amount of potential calculation time and reduce the computational complexity of our algorithms.

5.4.2 Real-Time Dispatching

In this section, we describe the dispatching algorithm DIS that is responsible for handling newly arrived trip requests. This algorithm is inspired by the approach by Ma et al. (2013) and Ma et al. (2015).

5.4.2.1 Algorithm Outline

Our dispatching procedure consists of three steps:

1. Vehicle selection – As the first step, we select a set of candidate vehicles that are potentially suited to insert the request into their route.
2. Vehicle sorting – As the second step, we sort all candidate vehicles according to the estimated suitability of inserting the request into the vehicles' routes.
3. Cheapest insertion – Lastly, we find the cheapest insertion among the candidate vehicles. We may abort the search prematurely after finding a feasible insertion.

These main steps are described in detail in the following sections.

5.4.2.2 Vehicle Selection

The objective of the vehicle selection step is to determine a set of candidate vehicles that are potentially well-suited to accommodate the new trip request. For this purpose, we utilize a grid-based index data structure that we will introduce in the following. Afterward, we describe how this data structure is used to determine candidate vehicles for a request.

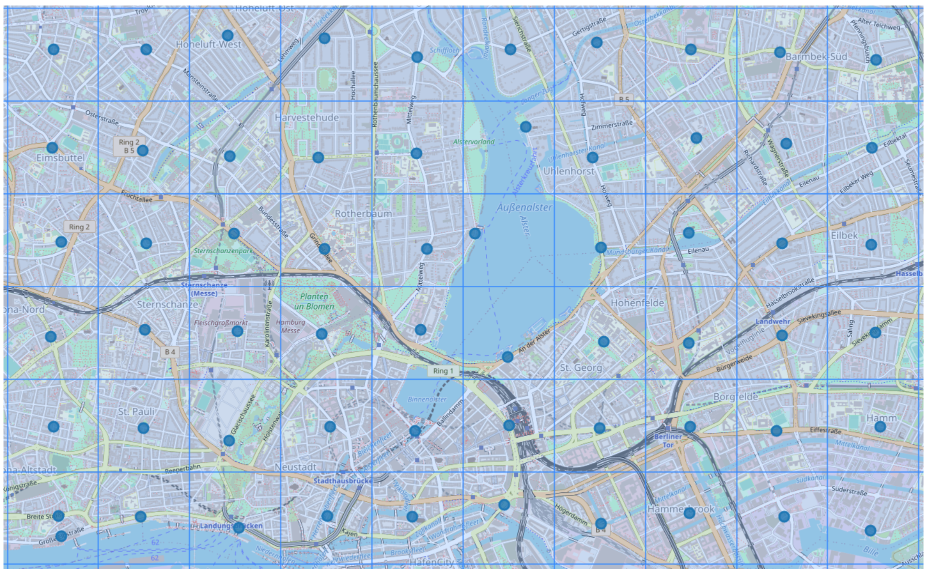


Figure 5.2: Grid partitioning with grid cell centers.

Grid-based partitioning It would be computationally intensive to use routing-based travel times to estimate whether a vehicle could potentially accommodate a new trip request. Hence, to speed up this calculation, we partition our area under study into a set of grid cells G_d. This is similar to the approach presented in Ma et al. (2013). An example of this grid partitioning is shown in Figure 5.2. The grid data structure fulfills two main purposes. Firstly, the grid may be utilized to estimate travel times between arbitrary locations. For this purpose, each grid cell is represented by a center c_g. This center corresponds to the node in the road network closest to the geometric centroid of the cell. We may now pre-compute a matrix of travel times between each pair of grid cell centers $c_g, c_h | g, h \in G_d$. These pre-computed travel times are used as fast travel time estimations. Assume that we have two locations i and j that lie within grid cells g and h respectively. We may use the pre-computed travel time between the grid cell centers as an estimate for the actual travel time between i and j. In the following, $tt^g_{i,j}$ denotes this grid-based travel time estimation. Secondly, using these estimations, the grid serves as a fast lookup method for vehicles. For each cell g in the grid, we store the set of vehicles currently located within the cell K_g. This way, when a new trip request arrives, we can quickly look up all vehicles in the vicinity of the requested pickup location. Please note that these grid-based travel time estimations are used at several points throughout the algorithm and not only during the vehicle selection step.

Vehicle candidate determination Given a trip request r, we first determine the grid cell g in which its pickup location p_r is situated. Subsequently, we select a subset of grid cells $H \subseteq G_d$ where g is reachable from $h \in H$ within the prescribed pickup time window of r. Hence, $t^{cur} + tt^{g}_{g,h} \leq l_{d,r}$ must hold true. Here, t^{cur} denotes the current time. Finally, we consider all vehicles as potential candidates for insertion that are currently situated in a grid cell in H. We denote this list of vehicles as K^c_r. This way, we may efficiently select all vehicles that are theoretically able to reach the pickup of r on time. We do not need to perform any expensive shortest-path calculations in this step as we rely on pre-calculated travel times.

5.4.2.3 Vehicle Sorting

After selecting the list of candidate vehicles K^c_r, we sort vehicles based on their suitability for trip request r. As actually evaluating insertions is the most expensive operation in our algorithm, we employ estimations to determine whether inserting a request r into the route of a vehicle k is promising. In the following, we first explain the calculation of the estimated insertion cost. Subsequently, we describe how we employ this estimation to sort our vehicles.

Estimated insertion cost For estimating the insertion cost of Request r into the route of vehicle k, we utilize the grid-based travel time estimations as outlined above in Section 5.4.2.2. The estimation of the insertion cost is outlined in Algorithm 1. First, in lines 1 and 2, we calculate a representative pickup time tp^{rep}_r as the middle of the time window $[e_{p,r}, l_{p,r}]$ and a representative drop-off time td^{rep}_r as the middle of the time window $[e_{d,r}, l_{d,r}]$. Subsequently, in lines 3 and 4, we determine the indices i^*, j^* in the route of k such that the representative pickup and drop-off lie in the interval between the end of the previous stop and the start of the following one. If no such position is found, the stops are inserted at the end of the route. The estimated insertion cost $\hat{\Delta}^+_{r,k}$ corresponds to the additional travel time as given by our grid-based estimations. Note that this estimated insertion cost

Algorithm 1: Estimated insertion

```
/* Estimates the cost of inserting r into the route of a vehicle k.    */
Input: Request r, vehicle k
Output: Estimated insertion cost $\hat{\Delta}^+_{r,k}$
1 $tp^{rep}_r$ ← Middle of time window $e_{p,r}, l_{p,r}$
2 $td^{rep}_r$ ← Middle of time window $e_{d,r}, l_{d,r}$
3 $i^*$ ← Index at which $de^k_{i-1} \leq tp^{rep}_r \wedge ar^k_i \geq tp^{rep}_r$ or end of route
4 $j^*$ ← Index at which $de^k_{i-1} \leq td^{rep}_r \wedge ar^k_i \geq td^{rep}_r$ or end of route
5 $\hat{\Delta}^+_{r,k}$ ← Grid-based insertion cost at positions $i^*, j^*$
6 return $\hat{\Delta}^+_{r,k}$
```

is also used in other phases of the algorithm, we will reference Algorithm 1 in the relevant sections.

Vehicle candidate sorting Given a vehicle $k \in K_r^c$ we utilize Algorithm 1 to calculate the estimated insertion cost $\hat{\Delta}^+_{r,k}$. Subsequently, we sort K_r^c in increasing order according to this estimated cost. This sorting procedure allows us to check the most promising vehicles for an insertion first and abort the search prematurely.

5.4.2.4 Cheapest Request Insertion

Finally, we iterate over the sorted vehicle candidates K_r^c and determine the cheapest feasible insertion of trip request r into the route of a vehicle $k \in K_r^c$. For this purpose, we utilize a cheapest insertion algorithm that we explain next. Afterward, we show how this algorithm is used to determine a vehicle's route into which r will be inserted.

Cheapest insertion algorithm One common operation in vehicle routing algorithms is the determination of the cheapest insertion of a request r into the route of a vehicle k. We utilize a straightforward cheapest insertion algorithm as given in Algorithm 2. As checking an insertion for feasibility and calculating the insertion cost is potentially expensive due to the necessary shortest path calculations, we use some simple pruning techniques to limit the range of insertion indices that are checked. In line 1, we determine a set of potentially feasible pickup indices $I_{r,k}$. This includes all positions i in the vehicle's current route for which the departure time at the previous stop is earlier than the latest pickup service, i.e. $de^k_{i-1} < l_{p,r}$. Additionally, the next stop $i+1$ is associated with a latest possible service l_{i+1}. An insertion position is potentially feasible, if $e_{p,r} < l_{i+1}$. A similar procedure is performed in line 3 for the drop-off stop. The drop-off must occur after the

Algorithm 2: Cheapest insertion

```
/* Finds the cheapest insertion of a request r into the route of a
   vehicle k.                                                      */
Input: Request $r$, vehicle $k$
Output: Cheapest feasible insertion indices $i^*, j^*$ with cost $\Delta^{+,*}_{r,k}$
1 $I_{r,k}$ ← Set of potentially feasible pickup indices
2 for $i \in I_{r,k}$ do
3     $J_{r,k,i}$ ← Set of potentially feasible drop-off indices
4     for $j \in J_{r,k,i}$ do
5         $\Delta^+_{r,k,i,j}$ ← Evaluate insertion at indices $i, j$
6         if Insertion is feasible and new best then
7             $i^*, j^*, \Delta^{+,*}_{r,k}$ ← $i, j, \Delta^+_{r,k,i,j}$
8 return $i^*, j^*, \Delta^{+,*}_{r,k}$
```

pickup and, in addition, the following conditions must hold: $de^k_{j-1} < l_{d,r}$ and $e_{d,r} < l_{j+1}$. For any potentially feasible combination i, j, we evaluate the insertion of r into the route and check its feasibility concerning the vehicle capacity, pickup time window, and ride time. If the insertion is feasible, we calculate the additional travel time incurred by this insertion as $\Delta^+_{r,k,i,j}$. We return the best feasible insertion positions i^*, j^* with the minimal insertion cost denoted as $\Delta^{+,*}_{r,k}$. Note that Algorithm 2 uses actual routing travel times instead of estimations. This ensures the feasibility of the found insertion.

Request insertion We utilize Algorithm 2 to determine a vehicle in K^c_r to which r will be assigned. For this purpose, we iterate over the sorted vehicle candidates $k \in K^c_r$ and evaluate the insertion into the vehicle's route. After checking k^{max} vehicles, we abort the search if a feasible insertion was found. Due to the prior sorting step, we have evaluated the most promising vehicles at this point. If no feasible insertion was found, we continue the search until either an insertion is found or we have iterated over all vehicles. As the final result of the dispatching phase, we have now determined a feasible vehicle route into which r is inserted. In case no feasible insertion was found, the request r is rejected.

5.4.3 Local Search Improvement

The second major algorithmic component of our vehicle routing approach is a local search algorithm LS. It uses the available computational time between the dispatching of incoming trip requests to improve the current routing plan. LS consists of several separate phases:

1. Inter-route search – Modifies two routes simultaneously.
    a) Inter-route move – Moves a single request from one vehicle to another.
    b) Inter-route swap – Swaps two requests between vehicles.
2. Intra-route search – Improves the route of a single vehicle.
    a) Intra-route stop move – Moves a single stop within a route.
    b) Intra-route request move – Moves the pickup and drop-off stops associated with a trip request within a route.

5.4.3.1 Common Components and Notation

Several elements are utilized repeatedly throughout our local search algorithm. In the following, we first introduce these common elements.

Request Queue The request queue contains the set of trip requests that are currently planned but have not yet been picked up. The queue is ordered in decreasing order by the travel time contribution of each request Δ^-_{r,k_r}. This corresponds to the travel time that would be saved if the trip request r were removed from its current vehicle k_r and is determined based on the travel times from our routing engine. The request queue is updated every time a trip request is inserted into or removed from a route. The reasoning for this is that we want to prioritize requests with a large Δ^-_{r,k_r} during our search, as they offer the most potential for improvement and we may not be able to evaluate all requests due to running time restrictions. Our two inter-route search neighborhoods each utilize a separate request queue RQ^m (move) and RQ^s (swap) to determine the next request for evaluation. The usage of separate queues is necessary due to the tabu mechanism described in the next paragraph and the separate time limits for both operators.

Tabu List In addition to the request queue, our inter-tour operators each use a tabu list denoted as TL^m (move) and TL^s (swap). These contain requests that have recently been evaluated without finding an improving move or swap respectively. For a given tabu interval t^{tabu}, a request on the tabu list is not evaluated for a potential move or swap again.

5.4.3.2 Inter-Route Search

The inter-route search modifies the routes of two vehicles simultaneously by trying to move a request from one vehicle to another or swapping two requests between vehicles while improving the overall vehicle travel times.

Inter-Route Move The inter-route move operator is outlined in Algorithm 3. It performs a first-improvement search until either the time limit T^m is reached or the request queue RQ^m is empty. We first select the next request r in the request queue. Subsequently, we iteratively determine a sorted set of potential vehicles to which r may be moved. Initially, in line 3, this set contains all vehicles except the vehicle to which r is currently assigned denoted as k_r. In line 4, we filter vehicles based on the travel time contribution of r and the estimated insertion cost $\hat{\Delta}^+_{r,k}$. We only consider vehicles for which $\hat{\Delta}^+_{r,k}$ is smaller or equal than Δ^-_{r,k_r} multiplied by a factor $f^{det} > 1.0$. In general, the actual cost of inserting r into the route of k must be smaller than Δ^-_{r,k_r} in order for a move to be worth it. As we only work with an estimated insertion in this step, we allow for some leeway by introducing the factor f^{det}. In line 5, we sort all remaining vehicles in increasing order according to $\hat{\Delta}^+_{r,k}$. This way, we evaluate vehicles first with a low estimated insertion cost and hence, the largest potential improvement. Subsequently, we iterate over the sorted vehicles and search for a feasible move. We utilize Algorithm 2 to determine the cheapest insertion of a request into a vehicle's route. If we find an improving move, i.e. one that is feasible and reduces the overall travel time, we perform this move in line 9. Subsequently, we update RQ^m and proceed with the next request. If we find no improving move for a given request, the request is added to TL^m.

Algorithm 3: First-improvement inter-route move

```
/* Performs a first improvement inter-route move search.                */
Input: Request queue $RQ^m$, tabu list $TL^m$, vehicles $K$
while $T^m$ is not reached and $RQ^m$ is not empty do
    $r \leftarrow$ next request in $RQ^m$
    $K^t \leftarrow K \setminus k_r$
    $K^t \leftarrow \{k \in K^t | \hat{\Delta}^+_{r,k} \leq f^{det} \cdot \Delta^-_{r,k_r}\}$
    $K^t \leftarrow$ sort $K^t$ by $\hat{\Delta}^+_{r,k}$
    for $k \in K^t$ do
        $m^* \leftarrow$ best move for $r, k_r, k$
        if $m^*$ is improving then
            perform $m^*$, update $RQ^m$ and break
    if no improving swap was found for $r$ then
        $TL^m \leftarrow TL^m \cup r$
```

Inter-Route Swap The inter-route swap search as outlined in Algorithm 4 behaves similarly to the inter-route move. In lines 3 – 5, in the same manner as before, we select the next request r_1 from the queue and the determine a sorted set of vehicles K^t. Now we iterate over vehicles $k_2 \in K^t$. Let R_{k_2} denote all requests that are assigned to k_2, but have not been picked up yet. These are the potential swap partners. We then find the best swap by performing a cheapest insertion of r_1 into k_2 and r_2 into k_{r_1} with Algorithm 2. If we find a swap that reduces the overall travel time, i.e. $\Delta^{+,*}_{r_1,k_2} + \Delta^{+,*}_{r_2,k_{r_1}} < \Delta^-_{r_1,k_{r_1}} + \Delta^-_{r,k_2}$, the swap is performed. Subsequently, we update RQ^s and continue with line 1. If no improving swap is found, r_1 is added to TL^s.

Algorithm 4: First-improvement inter-route swap

```
/* Performs a first improvement inter-route swap search.                */
Input: Request queue $RQ^s$, tabu list $TL^s$, vehicles $K$
while $T^s$ is not reached and $RQ^s$ is not empty do
    $r_1 \leftarrow$ next request in $RQ^s$
    $K^t \leftarrow K \setminus k_{r_1}$
    $K^t \leftarrow \{k \in K^t | \hat{\Delta}^+_{r_1,k} \leq f^{det} \cdot \Delta^-_{r_1,k_{r_1}}\}$
    $K^t \leftarrow$ sort $K^t$ by $\hat{\Delta}^+_{r_1,k}$
    for $k_2 \in K^t$ do
        for $r_2 \in R_{k_2}$ do
            $s^* \leftarrow$ best swap for $r_1, r_2, k_{r_1}, k_{r_2}$
            if $s^*$ is improving then
                perform $s^*$, update $RQ^s$ and break
    if no improving move was found for $r_1$ then
        $TL^s \leftarrow TL^s \cup r_1$
```

5.4.3.3 Intra-Route Search

The intra-route search improves the sequence of stops within a single route. It is applied to all vehicles whose routes have been modified since the last intra-route search iteration. These modifications may have either been performed by the dispatching algorithm or the inter-route search. We utilize two operators. The intra-route stop move removes a single stop and tries to re-insert it at another feasible position while reducing the overall travel time. The intra-route request move removes both stops belonging to a request that has not yet been picked up from the route and re-inserts them with Algorithm 2. Both operators are used in a first-improvement fashion, i.e. once an improving move is found, it is performed. The intra-route search has no time limit and is applied exhaustively until no further improvement is found. However, due to the small search space and the limited number of vehicles that need to be checked in each iteration, the running time is negligible.

5.5 Reactive Idle Vehicle Repositioning

While anticipatory routing and repositioning of idle vehicles is discussed in detail in Chapter 6, preliminary experiments have shown that the overall performance of our ride-sharing system suffers greatly when no repositioning at all is performed. This is mainly due to the fact that vehicles get stranded in low-demand areas and cannot reach arising trip requests in high-demand areas within their maximum waiting time. Hence, we introduce a simple repositioning approach to enable us to evaluate our vehicle routing algorithms. Otherwise, our performance metrics would be dominated by the lack of repositioning. We propose the usage of a reactive repositioning algorithm based on the work by Alonso-Mora et al. (2017a). Our approach is outlined in Algorithm 5. Given a rejected trip request r, we determine the idle vehicle k^* with the lowest travel time to p_r and reposition this vehicle to p_r. The intuitive reasoning behind this approach is that trip requests are highly spatially and temporally correlated. Hence, when we reject a trip request at a location, it is likely

Algorithm 5: Reactive repositioning

```
/* Repositions an idle vehicle (if available) to the pickup location of a
   rejected trip request.                                                  */
Input: Set of idle vehicles $K^{id}$, rejected trip request $r$
1 $tt^{min} \leftarrow \infty$
2 for $k \in K^{id}$ do
3     $tt_{k,p_r} \leftarrow$ travel time of $k$ to $p_r$
4     if $tt_{k,p_r} < tt_{min}$ then
5         $tt_{min} \leftarrow tt_{k,p_r}$
6         $k^* \leftarrow k$
7 Reposition $k^*$ to $p_r$
```

that additional trip requests will arise in the vicinity in the near future. Therefore, sending an idle vehicle to the pickup location of the request may ensure that we are capable of serving these future requests.

5.6 Computational Evaluation

In this section, we evaluate our vehicle routing algorithms on several real-world datasets. As a basis for our evaluation, we use the simulation-based framework as presented in Chapter 4. In the following, we first discuss our overall experimental design in Section 5.6.1 followed by the presentation of our computational results in Section 5.6.2.

5.6.1 Experimental Design and Setup

In our computational experiments, we investigate the performance of our vehicle routing approaches on several datasets. Our main goals are to:

- Evaluate the applicability of our vehicle routing approach on a diverse set of instances and show the robustness of our approach under varying circumstances.
- Show that our algorithm is capable of handling large-scale real-world scenarios in real-time.
- Assess the improvement gained by employing the local search improvement phase compared to only the dispatching heuristic.
- Investigate the impact of relevant settings and parameters such as, for instance, time windows and fleet sizes on the performance of our algorithms.

To evaluate these different aspects, we use four real-world datasets described in Section 5.6.1.1. Based on these datasets, we generate a set of problem instances as presented in Section 5.6.1.2. These instances are subsequently run with a wide range of different simulation and algorithm settings that are detailed in Sections 5.6.1.3 and 5.6.1.4. As mentioned in Chapter 4, our planning algorithms and simulation were implemented in C++. All computational studies were run on a computer with an Intel i7-6600U CPU and 20 GB of RAM.

5.6.1.1 Dataset Description

We evaluate our vehicle routing algorithms on multiple real-world datasets from the cities of Hamburg (HH)[1], New York City (NYC) (NYC Taxi and Limousine Commission 2022) and Chengdu (CH) (Didi Chuxing 2020). In addition, we build a fourth dataset based on the NYC dataset that only encompasses the area of Manhattan (MANH). All datasets contain

[1] Provided by PTV Group, Haid-und-Neu-Str. 15, 76131 Karlsruhe, Germany.

at least the information given in Table 5.3 – the coordinates of the desired pickup and drop-off location as well as the time and date at which the request enters the system. Some datasets originally provide the time at which the customer is picked up. In that case, we treat this as the arrival time of the request in the system.

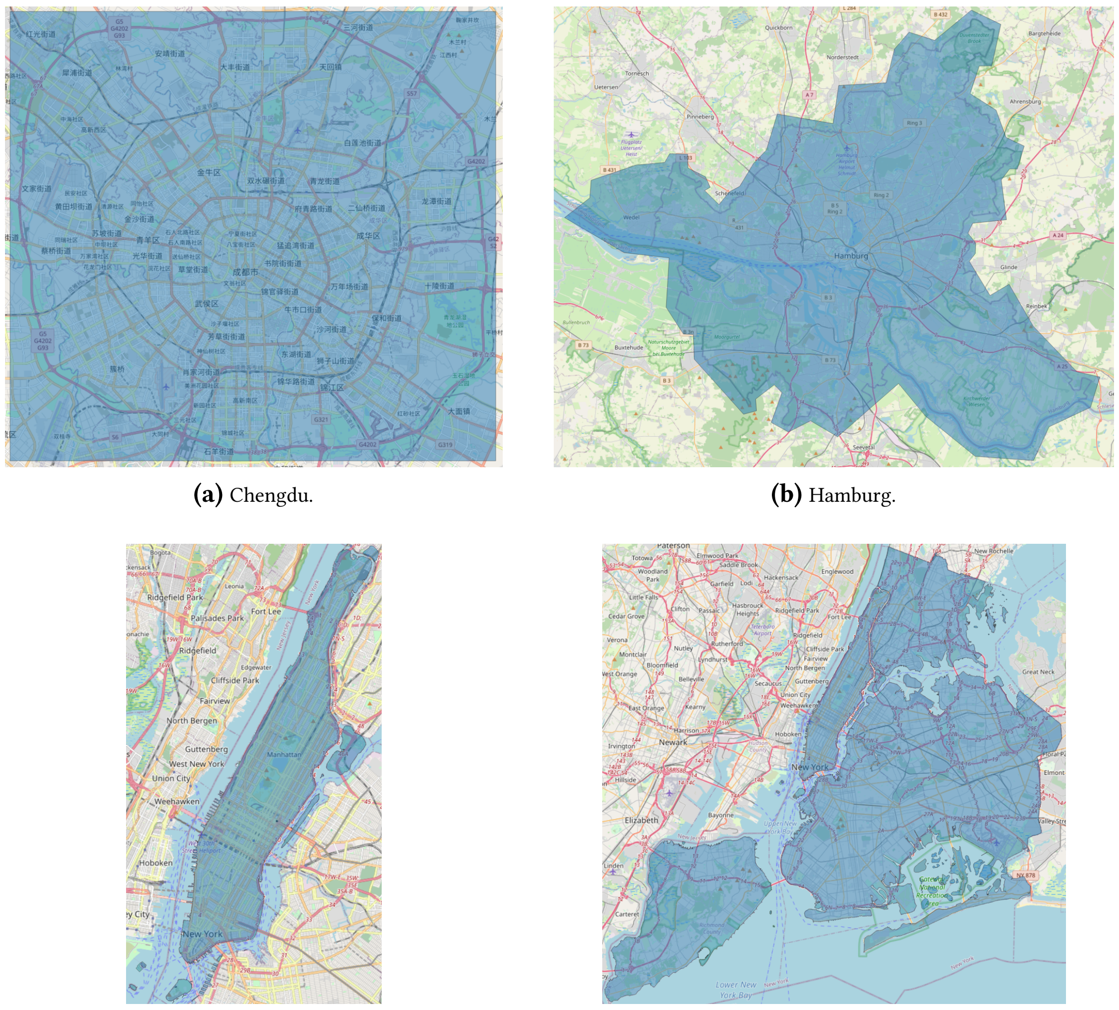
Table 5.3: Data structure of trip request datasets.

Pickup latitude	**Pickup longitude**	**Drop-off latitude**	**Drop-off longitude**	**Request datetime**
53.55762887	9.89704761	53.57196746	9.716359358	2019-03-16 14:42:00
53.46235765	9.98142367	53.44714752	9.958970077	2019-03-19 10:39:00
		...		

In addition to this data, we also need the number of passengers for each trip request in order to conduct our simulation studies. However, this information is only explicitly given for the NYC and MANH datasets. For these two datasets, we take the number of passengers as provided. We only consider trip requests with up to 2 passengers as this is common practice in ride-sharing services like UberPool (Uber 2021). Larger groups are generally requested to use ride-hailing services in which they occupy the complete vehicle for their trip and no sharing with other customers takes place. Hence, trip requests with three or more passengers are removed from the datasets. As the information concerning the passenger numbers is missing from the CH and HH datasets, we derive a probability distribution from the NYC dataset with a probability of 82.5 % for one passenger and 17.5 % for two passengers. The number of passengers for the CH and HH dataset is sampled from this distribution. The resulting enriched datasets may now be used as input for our simulation framework as described in Chapter 4.

For each dataset, we limit the geographical area in which trip requests may be situated. This is necessary due to the fact that some datasets contain outliers for which the desired pickup or drop-off location is far away from the area that we want to study. It is also common practice for ride-sharing services like MOIA (MOIA 2022a) to limit their service area to a region covering mostly densely populated areas close to the city center. The reasoning is that ride-sharing needs a certain level of demand density to be able to efficiently combine trip requests into shared routes. The service areas used in our simulation studies are depicted in Figure 5.3. Trip requests in the original data that lie outside these areas are removed from the datasets. For HH, MANH and NYC we restrict the covered areas to the administrative boundaries of the respective cities or boroughs. In the case of Chengdu, no such data was available to us. Hence, we restrict the area to a bounding box encompassing the outer ring road of Chengdu. In all cases, the vast majority of trip requests in the original data occur within the specified areas.

Given these datasets, we are able to evaluate our algorithms under diverse scenarios. In particular, we can evaluate the performance of our solution approach with regard to the following factors:



(a) Chengdu.

(b) Hamburg.

(c) Manhattan.

(d) New York City.

Figure 5.3: Dataset areas.

- Trip request demand – The total demand, as in the number of trip requests per day, varies between the datasets. While the CH, MANH and NYC datasets are relatively similar with around 300,000 trip requests per day, the HH dataset only contains ca. 15,000 trip requests per day. In addition, the demand density within each dataset varies between sub-regions. For instance, in the NYC dataset, roughly 80% of the demand occurs within Manhattan whereas the remaining areas have a comparatively low demand density.
- Covered area – The geographical area in which the trip requests in our datasets occur varies widely. The HH and NYC datasets cover relatively large areas that also encompass more sub-urban areas. In contrast, the MANH dataset, and, to a lesser extent, the CH dataset are focused on smaller, highly urbanized areas with a high population density.

5.6.1.2 Simulation Instances

Based on the datasets described above, we generate a set of simulation instances of which each instance covers a single day. The instances are divided into two groups: (1) preliminary test instances and (2) main evaluation instances. The former instances are used to perform preliminary experiments to determine adequate parameter settings while the latter are used for our main computational evaluations. The specific dates and number of trip requests for each instance are given in Table 5.4. For our main evaluation instances, we decided to seperately consider instances on a weekday (Wednesday) and on a weekend (Sunday). This way, we evaluate our algorithms under different demand patterns as the distribution of trip requests over the day varies significantly between weekdays and weekends as illustrated in Figure 5.4. Particularly for the HH, MANH and CH datasets, we may observe that on Sundays the demand at night is higher, but the peaks in the morning and evening that occur on weekdays are not as pronounced. We utilize a shorthand name for our instances consisting of the group (P – Preliminary, M – Main), the dataset and the weekday.

Table 5.4: Preliminary and main instances.

Group	Dataset	Weekday	Date	# trip requests	Name
Preliminary	CH	Wed	09 Nov 2016	224,219	P-CH-Wed
	HH	Wed	13 Mar 2019	16,158	P-HH-Wed
	MANH	Wed	09 Mar 2016	335,929	P-MANH-Wed
	NYC	Wed	09 Mar 2016	429,855	P-NYC-Wed
Main	CH	Wed	16 Nov 2016	239,037	M-CH-Wed
		Sun	20 Nov 2016	237,037	M-CH-Sun
	HH	Wed	20 Mar 2019	13,556	M-HH-Wed
		Sun	24 Mar 2019	10,669	M-HH-Sun
	MANH	Wed	16 Mar 2016	297,457	M-MANH-Wed
		Sun	20 Mar 2016	269,346	M-MANH-Sun
	NYC	Wed	16 Mar 2016	376,526	M-NYC-Wed
		Sun	20 Mar 2016	368,508	M-NYC-Sun

In addition to trip requests, we also need information concerning the vehicle fleet for our simulations. As we have no reliable real-world data regarding taxi fleets, we ran preliminary tests on the instances described above and determined a base fleet size for each dataset as given in Table 5.5. This fleet should be able to serve approximately 85 - 90 % of all trip requests. The assumption behind this choice of fleet size is that a real-world ride-sharing provider should be able to serve most trip requests entering the system. A low amount of rejections is allowed to account for excess demand during peak hours or trip requests in remote areas that are very difficult to reach. In Section 5.6.2.7, we also perform experiments with other fleet sizes.

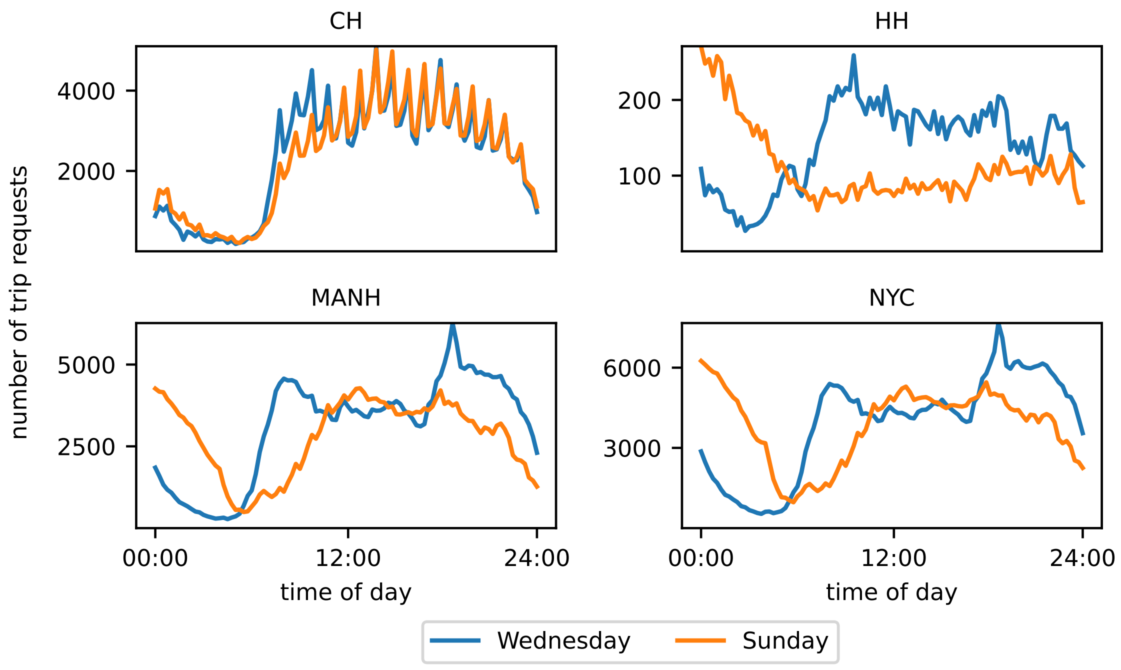


Figure 5.4: Distribution of trip requests over time on Wednesday and Sunday for the main instances of our datasets (aggregated in 15-minute intervals).

Table 5.5: Base fleet size per dataset.

	CH	**HH**	**MANH**	**NYC**
Fleet size	1260	90	670	1180

5.6.1.3 Simulation Settings

Table 5.6 summarizes the relevant settings for our simulation scenarios that are evaluated in the following sections. Unless noted otherwise the default values indicated in bold are used in our experiments. When it comes to parameters controlling customer time windows (w_r, L^{min}, m^{det}), we consider three different settings with short, medium and long time windows. These experiments are described in Section 5.6.2.6. Please note the simulation warm-up time of six hours. This means that we simulate six hours of service before beginning our collection of evaluation metrics. This ensures that we do not start with a system state in which all vehicles are idle which would distort our performance indicators. Therefore, when simulating an instance on 16 March 2016, the warm-up phase starts on 15 March 2016 at 18:00 and evaluation metrics are gathered after 00:00.

5.6.1.4 Algorithm Parameters

Table 5.7 summarizes all relevant parameters of our algorithms dispatching algorithm (DIS) and local search algorithm (LS). Most of our experiments are performed with and

Table 5.6: Simulation settings and potential values. Default values are indicated in **bold**.

Parameter	**Notation**	**Unit**	**Values**
Sharing allowed	-	-	**true**, false
Maximum waiting time	w_r	s	180, **300**, 600
Minimum allowed detour	L^{min}	s	100, **150**, 300
Maximum detour factor	m^{det}	-	1.33, **1.5**, 2
Vehicle factor	-	-	0.8, 0.9, **1.0**, 1.1, 1.2
Vehicle capacity	-	-	2, 3, **4**, 5, 6
Pre-booking times	pb_r	min	**0**, 15, 30, 60
Pre-booking percentage	-	%	**0**, 10, 25, 50
Service time	s_r	s	**10**
Warm-up time	-	h	**6**

Table 5.7: Algorithm parameters and potential values. Default values are indicated in bold.

Parameter	**Notation**	**Unit**	**Values**
Algorithm mode	-	-	DIS, **DIS+LS**
Grid cell side length	g_d^{size}	m	750, 1000, 1500, 2000, 3000
Vehicle limit	k^{max}	-	16, 32, 48, 96, ∞
Detour filter factor	f^{det}	-	1.0, 1.5, ∞
Tabu time	t^{tabu}	s	0, 60, 180, 300, ∞
Time limit move	T^m	ms	**80**, ∞
Time limit swap	T^s	ms	**120**, ∞
Repositioning	-	-	no, **yes**

without the local search phase in order to gauge its impact. These two algorithm modes are denoted as DIS and DIS+LS. Concerning the four parameters g_d^{size}, k^{max}, f^{det}, and t^{tabu}, we perform preliminary experiments in Section 5.6.2.1 to determine adequate values. Hence, no default values are given at this point. Regarding the time limits for the local search, we set the default values to a sum of 200 ms. This ensures that we can process all trip requests in real-time, even during peak demand hours. In Section 5.6.2.11, we also investigate scenarios in which we do not set a time limit to evaluate the potential performance gain. Unless noted otherwise, all evaluations are performed with the repositioning algorithm presented in Section 7. In Section 5.6.2.3, we evaluate scenarios without repositioning to illustrate its impact.

5.6.1.5 Performance Indicators

Table 5.8 summarizes the main performance indicators used throughout this section to assess the performance of our algorithms. The trip request rejection rate (*Rej*) is our main indicator for the overall system performance as we aim to maximize the number of accepted trip requests. *Wait* and *Ride* indicate the average waiting time and ride time

Table 5.8: Performance indicators.

KPI	Unit	Description
Rej	%	Trip request rejection rate
Wait	s	Avg. customer waiting time
Ride	s	Avg. customer ride time
TT^{v}	min	Avg. total vehicle travel time
TT^{v}_{req}	s	Avg. vehicle travel time per served trip request
RT	min	Total running time
RT^{dis}	min	Total running time for the dispatching algorithm
RT^{dis}_{r}	min	Avg. running time for dispatching one trip request
RT^{ls}	min	Total running time for the local search algorithm
RT^{re}	min	Total running time for the repositioning algorithm
RT^{o}	min	Total running time for other tasks

of a customer, respectively. The waiting time is the time between the earliest and actual pickup of a customer while the ride time is the time spent riding in the vehicle. These serve as ways to measure the impact on customer satisfaction. TT^{v} denotes the average total travel time of a single vehicle and serves as a proxy for the operational costs of operating the ride-sharing service. As the total vehicle travel time is strongly correlated with the number of served trip requests, we also report the average vehicle travel time per served request (TT^{v}_{req}). This corresponds to the total travel time of all vehicles divided by the number of served trip requests. Hence, it is normalized for the overall number of served trip requests and should indicate how efficiently we manage to plan the vehicle routes. Besides indicators concerning the solution quality, we also report the running times of our algorithm. RT denotes the total running time of a simulation run. This running time may be divided into the running time for DIS (RT^{dis}), the running time for LS (RT^{ls}), the running time for repositioning (RT^{re}), and the running time for other tasks, such as the simulation itself and maintaining the current state of the vehicle fleet (RT^{o}). In addition, we also report the average running time for dispatching a single trip request (RT^{dis}_{r}). This value should ideally be very small as it corresponds to the response time for an incoming trip request.

5.6.2 Computational Results

Throughout the following sections, we present the results of our computational experiments. We start by evaluating our algorithm parameters in Section 5.6.2.1 and determining a set of suitable parameter values for the remainder of this study. Subsequently, Section 5.6.2.2 discusses the running times of our algorithms. Section 5.6.2.3 presents the impact of our repositioning algorithm as proposed in Section 7. An overview of our computational results with default settings is presented in Section 5.6.2.4, while Sections 5.6.2.5 - 5.6.2.11 present a more detailed analysis concerning aspects such as vehicle sharing, time windows, fleet sizes, or pre-booking of trip requests.

5.6.2.1 Parameter Influence

In this section, we study the influence of several main parameters of our algorithms DIS and LS. For each parameter, we evaluate the different settings given in Table 5.9. Due to the involved computational effort, we do not perform a full grid search of all parameter combinations. Instead, we vary each parameter value individually setting the remaining parameters to their default starting value indicated in italic. The best-found settings that will be used for the remainder of this study are highlighted in bold. We mainly focus on the impact of each parameter value on the trip request rejection rate and the total running time.

Table 5.9: Algorithm parameters with default starting values denoted in *italic* and the best found values in **bold**.

Parameter	Notation	Unit	Values
Grid cell side length	g_d^{size}	m	***750***, 1000, 1500, 2000, 3000
Vehicle limit	k^{max}	-	16, 32, *48*, **96**, ∞
Tabu time	t^{tabu}	s	0, **60**, *180*, 300, ∞
Detour filter factor	f^{det}	-	**1.0**, *1.5*, ∞

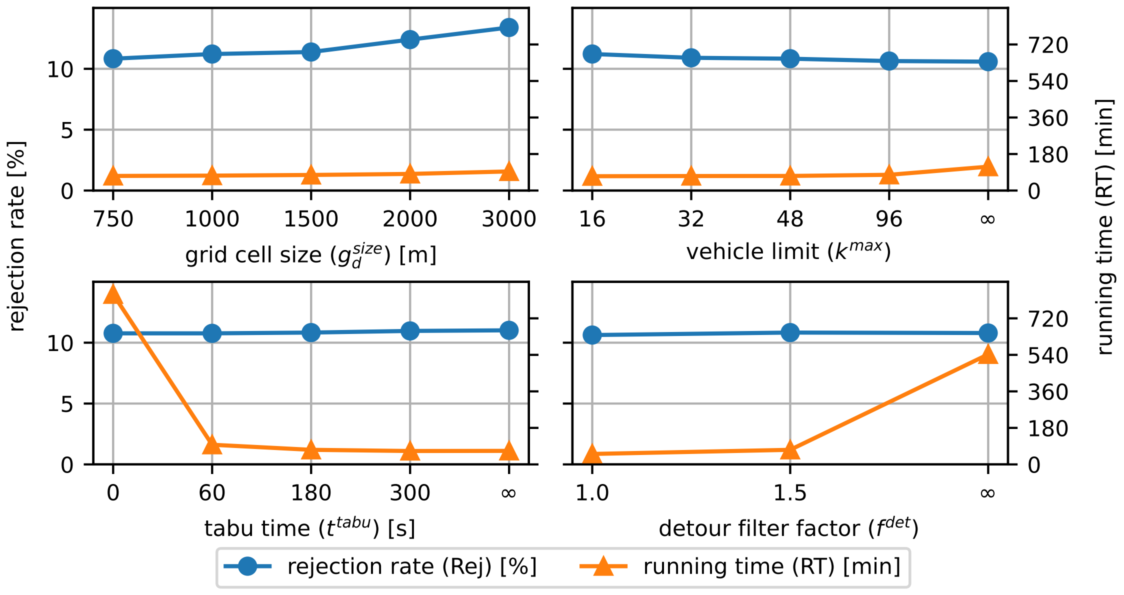


Figure 5.5: Influence of the grid cell side length (g_d^{size}), vehicle limit (k^{max}), detour filter factor (f^{det}), and tabu time (t^{tabu}).

The average results across our preliminary test instances are depicted in Figure 5.5. Our main findings may be summarized as follows:

- The grid cell side length g_d^{size} has almost no impact on the running time. This behavior is to be expected, as the grid calculation is a preprocessing step. A smaller grid size has a positive impact on the request rejection rates as both our travel time estimations and the lookup of potential vehicles for a trip request become more precise. Hence, we select a grid size of 750 m x 750 m. Smaller grid sizes are impractical due to the associated preprocessing time and the involved memory usage when covering large geographical areas.

- The vehicle limit k^{max} has a minor impact on the rejection rates. As expected, a larger number of evaluated vehicles correlates with reduced rejection rates. However, due to our vehicle sorting phase, this effect is not very pronounced as the most promising vehicles are evaluated first. In addition, going from a vehicle limit of 96 to evaluating all vehicles comes at a considerable runtime cost. Therefore, we set the vehicle limit to 96 as it presents a good trade-off between running time and rejection rate. Note that the different vehicle limits chosen for this experiment are multiples of the number of available threads on our machine, as vehicles are processed in parallel.
- The tabu time t^{tabu} has almost no impact on the rejection rate. Lower values are slightly better, but the effect is very minor. Setting the tabu time to 0 leads to a drastic jump in running time as each request is evaluated for potential moves and swaps in each iteration of the local search. In consequence, we choose a value of 60 seconds for the tabu time. Note that in our simulation experiments, t^{tabu} refers to the simulated time.
- Similarly, the detour filter factor has a very small impact on the rejection rate. In fact, all three settings lead to virtually identical results. In contrast, the running time of the algorithm may be reduced by setting f^{det} to a small value. Hence, we select a detour filter factor of 1.0.

5.6.2.2 Running Times

Table 5.10 shows a more detailed look at the running times of our algorithms DIS and LS. It reports the average running times for our main instances on each dataset. The main takeaway is that the running times are suitable for real-time usage in all evaluated scenarios. For each scenario, a timespan of 30 hours or 1800 minutes is simulated. This consists of the simulated 24 hours and the simulation warm-up time of 6 hours. Hence, even in the most computationally expensive scenarios, the simulation-based evaluation can be performed faster than in real-time by a large margin. For the larger datasets, the running times are heavily dominated by the dispatching and local search algorithms as these scale with the number of trip requests and the size of the vehicle fleet, whereas for the small instances in the HH dataset, the running times for the simulation and status management actually contribute a major portion. Looking at the dispatching running time for a single trip request RT_r^{dis}, we can see that even on the most challenging dataset (NYC), we still only need an average of roughly 10 ms to dispatch a single trip request. This facilitates fast response times for customers and leaves ample time for the local search to improve the routing plan.

Table 5.10: Overview of average running times. Rows denoted as "ALL" contain averages across all four datasets.

Data	**Mode**	**RT**^{dis} **[min]**	**RT**^{ls} **[min]**	**RT**^{re} **[min]**	**RT**^{o} **[min]**	**RT** **[min]**	**RT**$_{r}^{dis}$ **[ms]**
CH	DIS	22.91	0.00	0.88	3.97	27.76	6.91
	DIS+LS	24.17	33.54	1.14	11.66	70.51	7.22
HH	DIS	0.23	0.00	0.00	0.40	0.63	1.40
	DIS+LS	0.25	0.02	0.00	0.46	0.73	1.50
MANH	DIS	39.54	0.00	0.09	5.91	45.54	9.05
	DIS+LS	38.18	30.86	0.12	14.02	83.18	8.65
NYC	DIS	58.30	0.00	1.26	14.47	74.03	10.42
	DIS+LS	58.49	65.97	1.69	24.92	151.07	10.38
ALL	DIS	30.25	0.00	0.56	6.19	36.99	6.95
	DIS+LS	30.27	32.60	0.74	12.77	76.37	6.94

Table 5.11: Average results with and without repositioning. Rows denoted as "ALL" contain averages across all four datasets.

Data	**Repositioning**	**Rej** **[%]**	**RT** **[min]**	**RT**^{re} **[min]**	**RT**^{dis} **[min]**	**RT**^{ls} **[min]**
CH	no	61.35	35.52	0.00	19.65	9.64
	yes	10.80	70.51	1.14	24.17	33.54
HH	no	56.26	0.48	0.00	0.16	0.00
	yes	17.55	0.73	0.00	0.25	0.02
MANH	no	48.17	64.66	0.00	41.63	13.13
	yes	6.39	83.18	0.12	38.18	30.86
NYC	no	79.74	59.09	0.00	34.39	10.87
	yes	9.19	151.07	1.69	58.49	65.97
ALL	no	61.38	39.94	0.00	23.96	8.41
	yes	10.98	76.37	0.74	30.27	32.60

5.6.2.3 Impact of Repositioning

In Section 7, we introduced a reactive repositioning algorithm. The main reason for this was the fact that without any repositioning at all, the system performance deteriorates quickly. This effect is illustrated by the results in Table 5.11, where we show computational results with and without the proposed repositioning mechanism. As we can see, the rejection rate is increased drastically in scenarios where no repositioning is performed. This is due to the fact that vehicles are stuck in low-demand areas and cannot reach

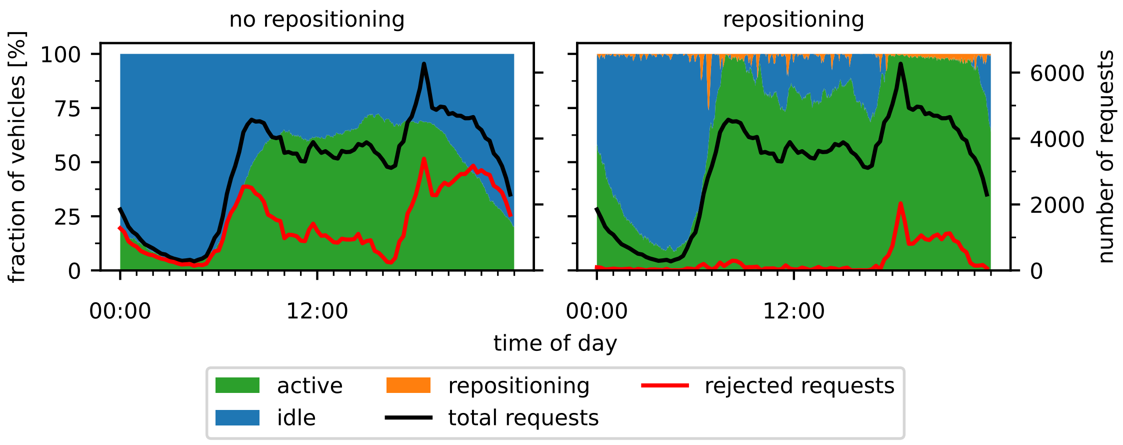


Figure 5.6: Vehicle states with and without repositioning for instance M-MANH-Wed.

incoming trip requests in other areas on time. Performing our repositioning mechanism remedies this and drastically reduces rejection rates. This effect can also be observed in the vehicle state chart in Figure 5.6. It shows the distribution of vehicle states over time for instance M-MANH-Wed. In addition, it depicts the number of total and rejected trip requests. In the scenario without repositioning, a large number of trip requests have to be rejected although a major fraction of the vehicle fleet is idle. In contrast, when using the reactive repositioning mechanism, the complete vehicle fleet is utilized during peak demand times. As seen by the results in Table 5.11, running times are increased when performing repositioning. However, this is not mainly due to the running time of the repositioning algorithm (RT^{re}). Rather, the running times for dispatching and local search increase. The former is due to the tendency that more vehicles are available close to a new trip request. Hence, we can check more vehicles for insertion in our dispatching algorithm. The latter is due to the fact that we serve more trip requests and consequently the search space in our local search phase is larger. All other evaluations in this section are performed with our repositioning mechanism activated.

5.6.2.4 Results on Default Scenarios

Table 5.12 summarizes our computational results with the two algorithm modes DIS and DIS+LS on the scenarios with default settings as presented in Section 5.6.1.3. On average, activating the local search leads to a reduction in the number of rejected requests of 5.2 %. Besides this, we observe minor improvements in the customer waiting and ride times. There is one exception in the HH dataset, where results with the local search are actually slightly worse than without. We believe that this is due to the low demand density in Hamburg combined with the relatively narrow time windows in the base scenario. In general, the results in Section 5.6.2.6 will show that the local search performs better in scenarios with more flexible time windows as this leaves a larger search space for the local search operators. Nevertheless, when it comes to the vehicle travel time measures TT^{v} and TT^{v}_{req}, we can see that the local search consistently plans more efficient vehicle routes

Table 5.12: Aggregated results with and without local search. Rows denoted as “ALL” contain averages across all four datasets.

Data	**Mode**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT^v_{req} [s]**	**RT [min]**
CH	DIS	11.67	**200.47**	595.64	965.25	366.96	27.76
	DIS+LS	**10.80**	200.52	**593.13**	**929.69**	**350.00**	70.51
HH	DIS	**17.42**	**174.94**	**475.59**	855.58	462.76	0.63
	DIS+LS	17.55	174.95	476.19	**842.71**	**456.72**	0.73
MANH	DIS	7.36	227.01	295.19	1071.16	164.34	45.54
	DIS+LS	**6.39**	**222.96**	**293.07**	**1028.68**	**156.16**	83.18
NYC	DIS	9.86	218.73	378.00	1086.75	229.16	74.03
	DIS+LS	**9.19**	**216.69**	**375.70**	**1037.41**	**217.17**	151.07
ALL	DIS	11.58	205.29	436.11	994.69	305.81	36.99
	DIS+LS	**10.98**	**203.78**	**434.52**	**959.62**	**295.01**	76.37

Table 5.13: Results with and without ride-sharing. Rows denoted as “ALL” contain averages across all four datasets.

Data	**Ride-Sharing**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT^v_{req} [s]**	**TT^{dir} [s]**
CH	no	32.05	240.30	**430.75**	1085.75	536.59	430.75
	yes	**10.80**	**200.52**	593.13	**929.69**	**350.00**	445.35
HH	no	26.54	188.54	**369.19**	968.46	588.55	369.19
	yes	**17.55**	**174.95**	476.19	**842.71**	**456.72**	381.02
MANH	no	32.82	264.64	**194.68**	1220.71	258.36	194.68
	yes	**6.39**	**222.96**	293.07	**1028.68**	**156.16**	198.54
NYC	no	30.24	253.21	**252.96**	1252.69	341.38	252.96
	yes	**9.19**	**216.69**	375.70	**1037.41**	**217.17**	267.94
ALL	no	30.41	236.67	**311.90**	1131.90	431.22	311.90
	yes	**10.98**	**203.78**	434.52	**959.62**	**295.01**	323.21

for all datasets. On average, TT^v_{req} is reduced by 3.5 % compared to the setting without local search.

5.6.2.5 Impact of Ride-Sharing

In Table 5.13, we show results with and without ride-sharing. Note that the last column denoted TT^{dir} shows the average direct travel time of served trip requests, i.e. the travel time between their pickup and drop-off location. As expected, the rejection rate is significantly higher without ride-sharing, as vehicles cannot transport multiple customer groups at once. In addition, the waiting time of customers is reduced when ride-sharing is allowed, as it is easier to insert a trip request into a vehicle route. Without ride-sharing, customers must wait until a vehicle has finished serving its current trip request. In contrast, the ride times are naturally higher with ride-sharing as we allow for detours compared to the direct travel time. However, the planned vehicle routes are more efficient as illustrated by TT^{v}_{req}. This illustrates that ride-sharing can be an improvement compared to ride-hailing or taxi services when it comes to traffic congestion as vehicles spend less time on the road. In three out of four datasets, TT^{v}_{req} is even lower than TT^{dir}. In these cases, ride-sharing leads to a reduced traffic load compared to a setting in which all customers would drive directly with their own vehicle. On average, the time spent by vehicles on the road may be reduced by 8.7 % by using ride-sharing. This illustrates the potential for ride-sharing services to help tackle traffic congestion problems in cities. However, the fact that for the HH dataset no improvement compared to TT^{dir} is achieved also shows that a sufficiently dense demand is paramount to achieving this benefit of ride-sharing services.

Table 5.14: Time window settings.

	Unit	Short	Medium	Long
w_r	s	180	300	600
L^{min}	s	100	150	300
m^{det}	-	1.33	1.5	2.0

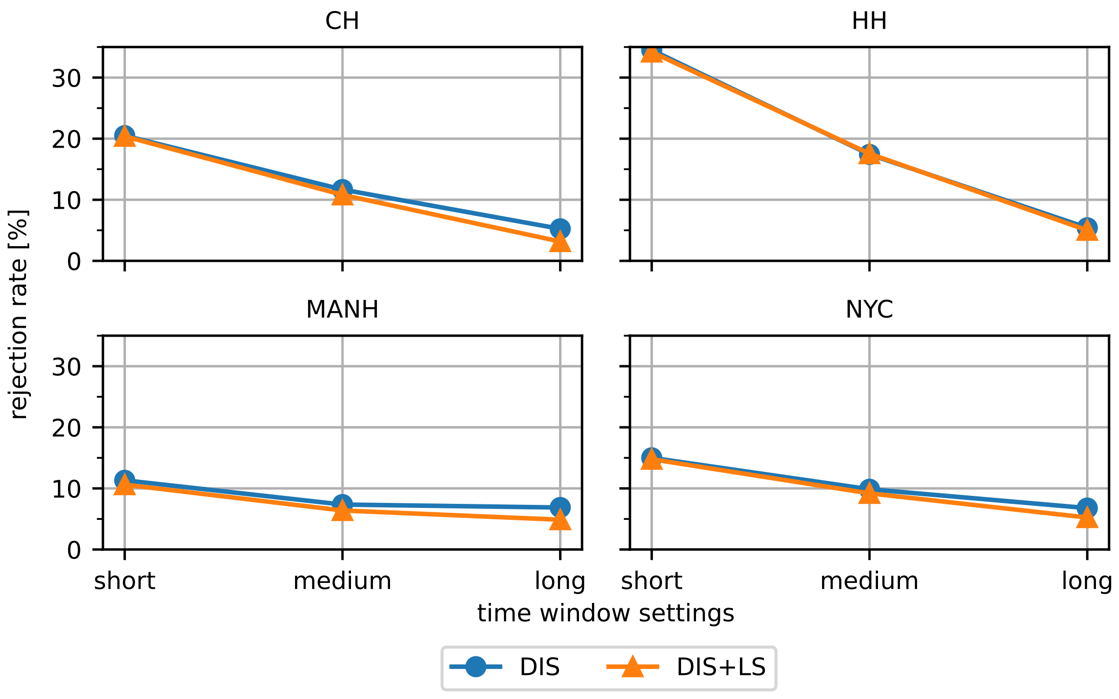


Figure 5.7: Rejection rates for different time window settings.

5.6.2.6 Time Windows

In this section, we take a look at the impact of customer time windows on our vehicle routing algorithms. We consider three different time window settings: short, long and medium time windows. The specific values for the relevant parameters controlling the maximum waiting time (w_r), the minimum additional detour (L^{min}) and the maximum detour factor (m^{det}) are given in Table 5.14.

Table 5.15: Results for different time window settings.

Data	**Time Windows**	**Mode**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT^v_{req} [s]**	**RT [min]**
CH	short	DIS	20.49	107.75	**528.87**	959.52	405.28	17.90
		DIS+LS	**20.39**	**109.57**	531.01	**925.87**	**390.55**	42.76
	medium	DIS	11.67	**200.47**	595.64	965.25	366.96	27.76
		DIS+LS	**10.80**	200.52	**593.13**	**929.69**	**350.00**	70.51
	long	DIS	5.24	460.50	738.71	969.28	343.49	45.62
		DIS+LS	**3.16**	**452.77**	**716.70**	**915.97**	**317.61**	155.52
HH	short	DIS	34.50	**100.36**	**423.21**	807.33	549.75	0.43
		DIS+LS	**34.18**	100.39	425.85	**797.60**	**540.65**	0.51
	medium	DIS	**17.42**	**174.94**	**475.59**	855.58	462.76	0.63
		DIS+LS	17.55	174.95	476.19	**842.71**	**456.72**	0.73
	long	DIS	5.41	389.89	603.48	855.10	404.05	1.18
		DIS+LS	**5.00**	**383.23**	**596.53**	**822.24**	**386.93**	1.46
MANH	short	DIS	11.32	121.80	262.61	1100.22	176.36	29.49
		DIS+LS	**10.62**	**120.95**	**262.25**	**1072.30**	**170.51**	48.76
	medium	DIS	7.36	227.01	295.19	1071.16	164.34	45.54
		DIS+LS	**6.39**	**222.96**	**293.07**	**1028.68**	**156.16**	83.18
	long	DIS	6.87	507.33	368.93	1068.39	163.08	83.22
		DIS+LS	**4.88**	**490.81**	**356.65**	**1004.17**	**150.04**	216.53
NYC	short	DIS	15.01	**117.74**	**336.09**	1119.60	250.39	49.66
		DIS+LS	**14.78**	118.17	337.12	**1082.99**	**241.58**	90.32
	medium	DIS	9.86	218.73	378.00	1086.75	229.16	74.03
		DIS+LS	**9.19**	**216.69**	**375.70**	**1037.41**	**217.17**	151.07
	long	DIS	6.78	496.43	469.16	1072.59	218.67	138.77
		DIS+LS	**5.23**	**487.28**	**454.67**	**1004.97**	**201.58**	376.65
ALL	short	DIS	20.33	**111.91**	**387.70**	996.67	345.45	24.37
		DIS+LS	**19.99**	112.27	389.06	**969.69**	**335.82**	45.59
	medium	DIS	11.58	205.29	436.11	994.69	305.81	36.99
		DIS+LS	**10.98**	**203.78**	**434.52**	**959.62**	**295.01**	76.37
	long	DIS	6.08	463.54	545.07	991.34	282.32	67.20
		DIS+LS	**4.57**	**453.52**	**531.14**	**936.84**	**264.04**	187.54

Table 5.15 shows our computational results. The main conclusions are as follows. The improvement in the number of rejected trip requests achieved by the local search algorithm grows larger as the time windows are increased. For the long time window setting, we see an average improvement of 24.8 %. This is mainly due to the fact that the short and medium time windows leave little room for improvement as the search space is highly constrained by the narrower time windows. With longer time windows, the local search has a larger search space to explore and is capable of exploiting this. The impact of the time window length on the rejection rates is also illustrated in Figure 5.7. As expected, this improvement in rejection rate comes at the cost of an increased running time. However, the overall running time of our simulations is still significantly faster than real-time. Another observation is that the local search algorithm is able to exploit larger time windows to build more efficient vehicle routes. This may be seen by the decrease in TT^{v}_{req} when comparing the two modes DIS and DIS+LS. For the long time windows, the lowest absolute values are achieved and the improvement of DIS+LS compared to DIS is also the largest at 6.5 %, compared to 3.5 % and 2.8 % with medium and short time windows respectively.

5.6.2.7 Fleet Sizes

We evaluate different vehicle fleet sizes by applying a factor ranging from 0.8 - 1.2 to our base fleet size as determined in Section 5.6.1.2. Aggregated results for vehicle factors of 0.8, 1.0 and 1.2 are presented in Table 5.16. One noteworthy result is that LS manages to substantially improve the results for smaller fleet sizes whereas the performance between

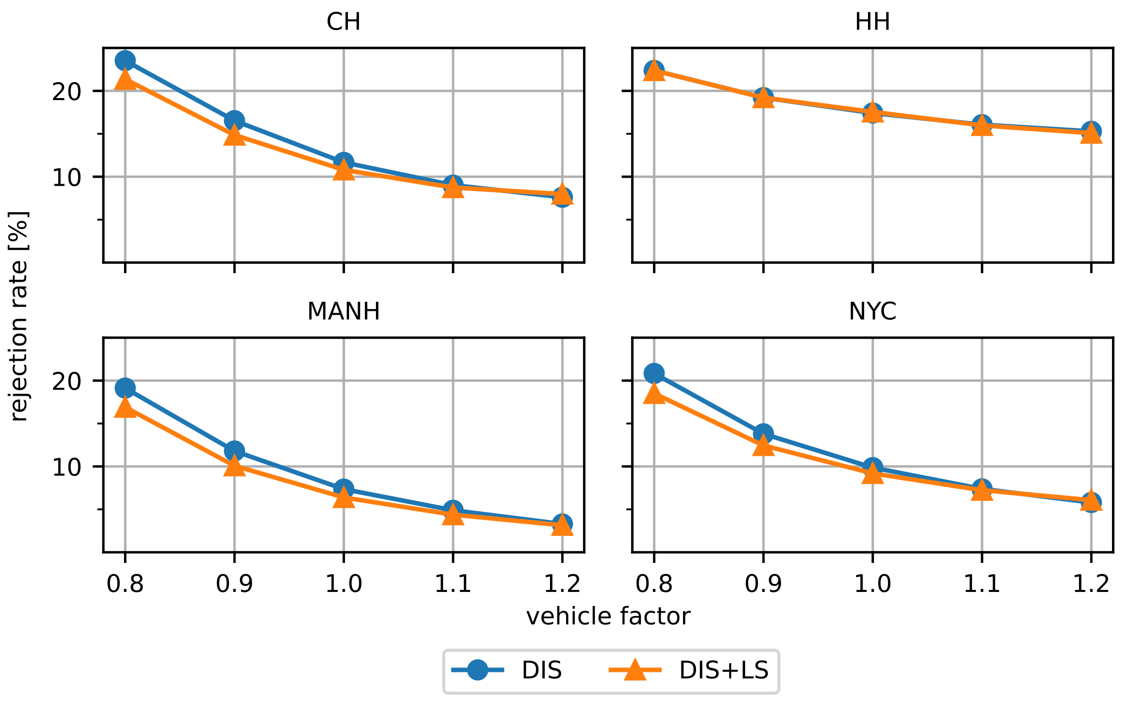


Figure 5.8: Rejection rates for different fleet sizes.

Table 5.16: Results with different fleet sizes.

Data	**Vehicle Factor**	**Mode**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT$^v_{req}$ [s]**	**RT [min]**
CH	0.8	DIS	23.52	**205.78**	611.69	1068.16	375.20	28.29
		DIS+LS	**21.36**	205.92	**606.96**	**1050.69**	**358.92**	57.78
	1.0	DIS	11.67	**200.47**	595.64	965.25	366.96	27.76
		DIS+LS	**10.80**	200.52	**593.13**	**929.69**	**350.00**	70.51
	1.2	DIS	**7.61**	**195.12**	591.08	833.28	363.43	28.17
		DIS+LS	7.98	196.60	**589.96**	**790.15**	**346.01**	76.47
HH	0.8	DIS	22.38	**177.98**	477.80	1013.92	467.32	0.53
		DIS+LS	**22.37**	178.02	**477.43**	**1001.72**	**461.39**	0.65
	1.0	DIS	**17.42**	**174.94**	**475.59**	855.58	462.76	0.63
		DIS+LS	17.55	174.95	476.19	**842.71**	**456.72**	0.73
	1.2	DIS	15.28	172.06	**475.47**	723.48	458.02	0.69
		DIS+LS	**15.08**	**171.94**	476.91	**711.75**	**449.70**	0.82
MANH	0.8	DIS	19.12	235.94	299.49	1182.11	166.23	49.43
		DIS+LS	**16.90**	**232.23**	**297.42**	**1164.89**	**159.41**	77.31
	1.0	DIS	7.36	227.01	295.19	1071.16	164.34	45.54
		DIS+LS	**6.39**	**222.96**	**293.07**	**1028.68**	**156.16**	83.18
	1.2	DIS	3.29	223.39	294.06	931.22	164.16	41.89
		DIS+LS	**3.15**	**220.28**	**292.23**	**882.46**	**155.32**	86.36
NYC	0.8	DIS	20.83	224.37	385.26	1226.03	235.57	76.97
		DIS+LS	**18.51**	**221.75**	**381.25**	**1198.45**	**223.67**	127.10
	1.0	DIS	9.86	218.73	378.00	1086.75	229.16	74.03
		DIS+LS	**9.19**	**216.69**	**375.70**	**1037.41**	**217.17**	151.07
	1.2	DIS	**5.80**	216.70	377.09	939.47	227.57	73.43
		DIS+LS	6.07	**214.80**	**375.02**	**885.41**	**215.11**	167.43
ALL	0.8	DIS	21.46	211.02	443.56	1122.56	311.08	38.81
		DIS+LS	**19.79**	**209.48**	**440.77**	**1103.94**	**300.85**	65.71
	1.0	DIS	11.58	205.29	436.11	994.69	305.81	36.99
		DIS+LS	**10.98**	**203.78**	**434.52**	**959.62**	**295.01**	76.37
	1.2	DIS	**8.00**	201.82	434.43	856.86	303.30	36.05
		DIS+LS	8.07	**200.91**	**433.53**	**817.44**	**291.54**	82.77

DIS and DIS+LS is rather similar for a larger fleet. With factor 0.8, the average improvement in rejection rates is 1.67 percentage points, whereas with factor 1.2, we actually obtain slightly better results without the local search. This aspect may also be observed in Figure 5.8 where we additionally show the rejection rates for vehicle factors 0.9 and 1.1. We assume that the reason for this behavior is that the local search builds more efficient vehicle routes as observed by the vehicle travel times per served request TT$^v_{req}$. Hence, with LS we are able to serve more customers given the same amount of vehicle resources.

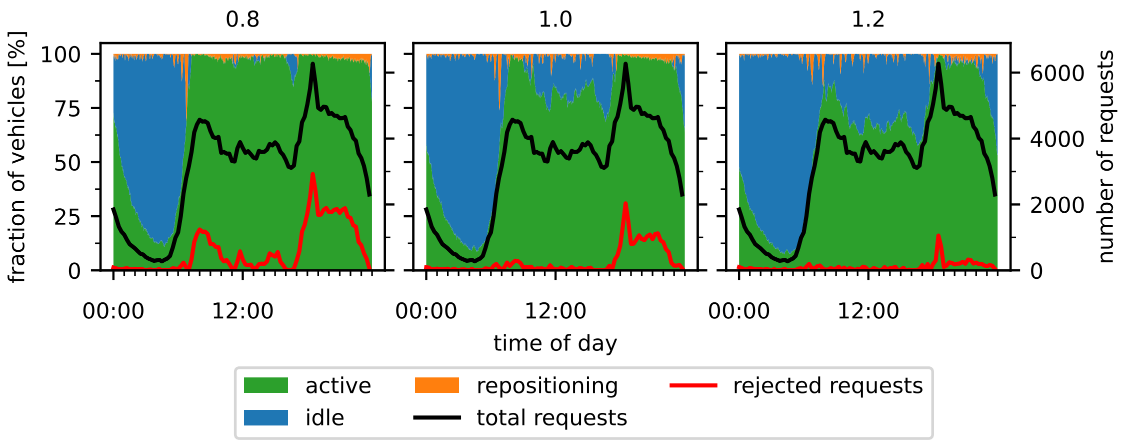


Figure 5.9: Vehicle states with different fleet factors for scenario M-MANH-Wed and DIS+LS.

This aspect grows less important in scenarios with a larger fleet. In these scenarios, trip requests are not mainly rejected due to the complete fleet being occupied. Instead, we reject trip requests in remote areas that are difficult to reach. The main constraining factor is no longer the fleet size in scenarios with factor 1.2, which is further illustrated in Figure 5.9 which shows the state of the vehicle fleet for scenario M-MANH-Wed and different fleet factors. We may see that with lower fleet factors, the complete vehicle fleet is occupied during peak demand times. With factor 1.2, this is no longer the case. To enable these available vehicles to serve the trip requests that are currently rejected, we would need anticipatory routing or repositioning approaches. This is discussed in detail in Chapter 6. We believe that the slightly worse performance of DIS+LS compared to DIS with the largest fleet size is not due to structural reasons, but rather due to differences in the created vehicle routes. Both algorithms should perform very similarly regarding the rejection rate in scenarios with excess vehicles. We may also observe that DIS+LS still manages to reduce TT^{v}_{req}.

5.6.2.8 Vehicle Capacity

Besides the size of the vehicle fleet, we also evaluate the impact of the vehicle's capacities. We perform experiments with a capacity of 2 – 6. The results for capacities 2, 4, and 6 are given in Table 5.17. Additionally, Figure 5.10 shows the rejection rates for all capacities. Similarly to the experiments with different fleet sizes, we can again observe that the improvement of DIS+LS over DIS is greatest with the smallest vehicle capacity (0.99 percentage points at capacity 2 versus 0.49 percentage points at capacity 6). We believe that the reason for this is the greater importance of efficient routes in scenarios where the total capacity of the fleet is a severely constraining factor. As expected, larger capacities lead to an overall decreased rejection rate and a lower TT^{v}_{req}. However, we see diminishing returns here and increasing the capacity beyond 4 only leads to minor improvements. It becomes increasingly difficult to exploit this capacity, mainly due to the time window

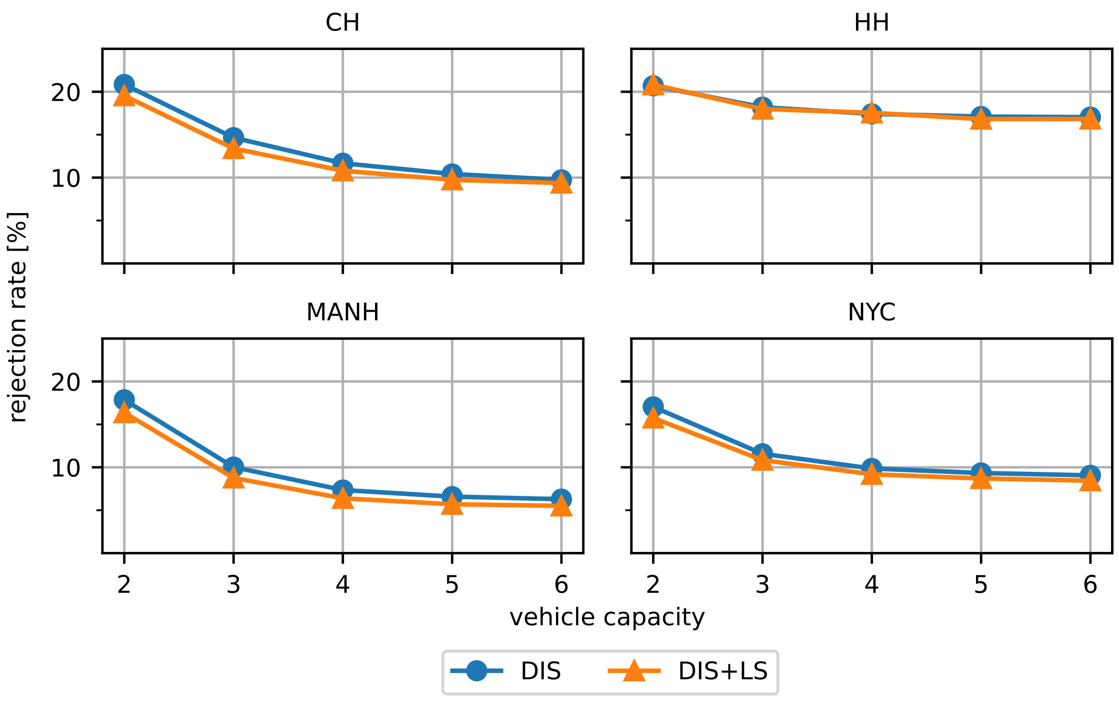


Figure 5.10: Rejection rates for different capacities.

constraints of trip requests. In use cases with larger time windows or passenger groups of more than 2 people, the benefit of higher vehicle capacities may be more pronounced.

5.6.2.9 Pre-Booked Trip Requests

One aspect that is rarely studied in literature concerning ride-sharing services, is the inclusion of pre-booked trip requests, i.e. trip requests that are known in advance and desire service at some point in the future. We believe that this is a relevant use case for ride-sharing providers. Depending on the fraction of pre-booked requests and the time interval between the submission of a request and its desired service, the degree of dynamism of the problem may change substantially. We evaluate the impact of both the pre-booking time and the pre-booking probabilities separately. Figure 5.11 shows the rejection rates in scenarios where the percentage of pre-booked trip requests varies between 0 % and 50 %. The pre-booking time pb_r is set to a fixed value of 15 minutes. The first thing to note is that DIS performs worse with a larger fraction of pre-booked trip requests. This is due to the fact that by itself it is not well suited to handle trip requests that are not served immediately as it greedily inserts each trip request at the best found route position regardless of when the desired start of service is. This aspect may be offset by using LS which benefits from having a higher percentage of pre-booked trip requests. These requests provide a larger search space for the local search as trip requests can be flexibly moved until their start of service.

Table 5.17: Results with different vehicle capacities.

Data	**Vehicle Capacity**	**Mode**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT^v_{req} [s]**	**RT [min]**
CH	2	DIS	20.84	221.44	547.82	1044.40	443.07	21.45
		DIS+LS	**19.54**	**220.75**	**538.49**	**1027.67**	**428.91**	67.14
	4	DIS	11.67	**200.47**	595.64	965.25	366.96	27.76
		DIS+LS	**10.80**	200.52	**593.13**	**929.69**	**350.00**	70.51
	6	DIS	9.73	196.11	600.55	942.33	350.55	28.71
		DIS+LS	**9.37**	**196.05**	**600.46**	**899.67**	**333.33**	70.68
HH	2	DIS	**20.69**	**180.91**	**442.94**	909.15	511.43	0.41
		DIS+LS	20.82	179.96	441.41	**899.82**	**507.36**	0.53
	4	DIS	**17.42**	**174.94**	**475.59**	855.58	462.76	0.63
		DIS+LS	17.55	174.95	476.19	**842.71**	**456.72**	0.73
	6	DIS	17.04	**173.30**	479.62	848.29	456.91	0.44
		DIS+LS	**16.83**	173.39	**479.56**	**832.89**	**447.56**	0.58
MANH	2	DIS	17.86	252.77	263.07	1176.15	203.56	34.44
		DIS+LS	16.37	249.89	257.69	**1161.91**	**197.50**	77.84
	4	DIS	7.36	227.01	295.19	1071.16	164.34	45.54
		DIS+LS	6.39	222.96	293.07	**1028.68**	**156.16**	83.18
	6	DIS	6.29	221.74	299.63	1047.12	158.80	48.65
		DIS+LS	5.50	217.29	299.12	**998.05**	**150.07**	87.61
NYC	2	DIS	17.04	240.96	334.53	1205.26	276.21	52.98
		DIS+LS	15.74	238.72	327.32	**1180.09**	**266.27**	137.97
	4	DIS	9.86	218.73	378.00	1086.75	229.16	74.03
		DIS+LS	9.19	216.69	375.70	**1037.41**	**217.17**	151.07
	6	DIS	9.06	213.70	384.41	1060.83	221.74	74.20
		DIS+LS	8.44	210.27	384.46	**1009.59**	**209.62**	148.78
ALL	2	DIS	19.11	224.02	397.09	1083.74	358.57	27.32
		DIS+LS	**18.12**	**222.33**	**391.23**	**1067.37**	**350.01**	70.87
	4	DIS	11.58	205.29	436.11	994.69	305.81	36.99
		DIS+LS	**10.98**	**203.78**	**434.52**	**959.62**	**295.01**	76.37
	6	DIS	10.53	201.21	441.05	974.64	297.00	38.00
		DIS+LS	**10.04**	**199.25**	**440.90**	**935.05**	**285.15**	76.91

In Figure 5.12 we show the rejection rates with different pre-booking times. The pre-booking probability is set to a fixed 25 % while the times are either 15 minutes or equally distributed between 15 − 30 or 15 − 60 minutes respectively. As before, when comparing the two algorithm modes DIS and DIS+LS, we may see that the gap in rejection rate between them increases as pre-booking is introduced. However, there is another trend in the results as both algorithm modes start to perform worse as the pre-booking time is increased. While DIS+LS benefits from pre-booked trip requests with a pre-booking

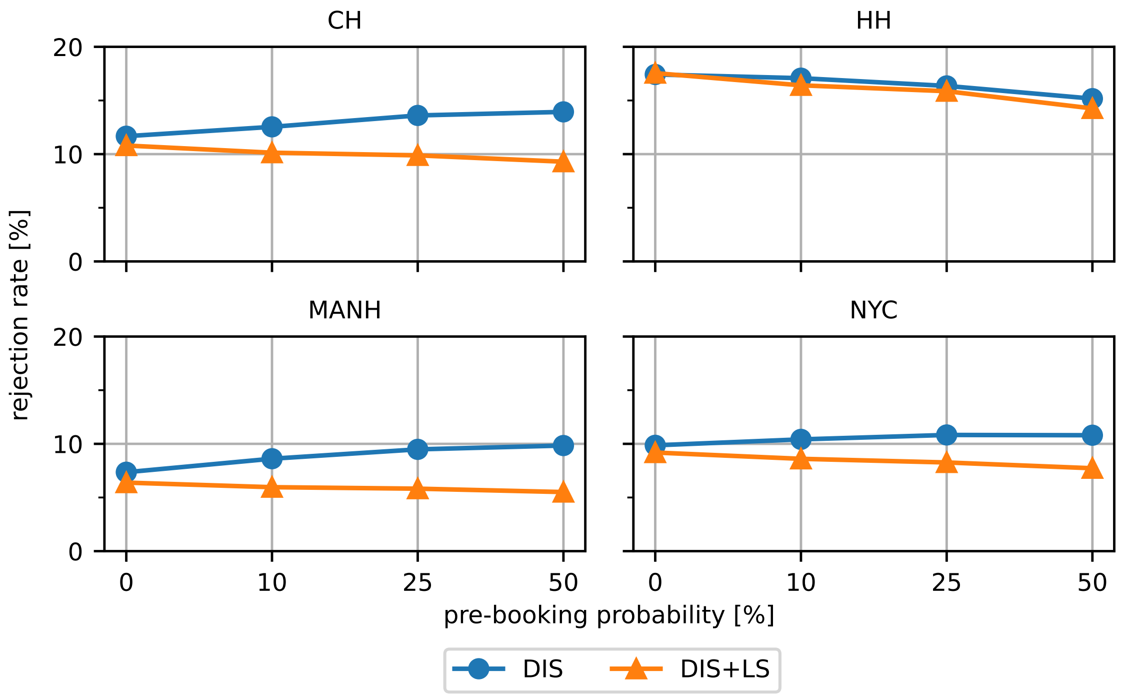


Figure 5.11: Rejection rates for different pre-booking probabilities.

time of 15 minutes, this is no longer always the case with longer pre-booking times. This shows that it is not beneficial to plan trip requests with a requested pickup time of more than 15 minutes in the future as this merely pollutes the routing plan with stops that do not need to be served immediately. We believe that this is currently a research gap when it comes to algorithms for the VRPDRS. To our knowledge, most current approaches are only evaluated on purely dynamic instances and as the results in this section have shown, this does not necessarily translate to good results on instances with less dynamism. For use cases where a large portion of trip requests is known in advance, it can be useful to implement specific solution approaches, e.g. combine static vehicle routing algorithms on pre-booked requests with dynamic approaches for the dynamically arriving requests.

5.6.2.10 Impact of Different Weekdays

So far, we only looked at aggregated results that combine both weekday scenarios (Wednesday and Sunday). This was mainly due to the fact that the weekday and particularly the different demand patterns had no noticeable impact on the algorithm performance. Figure 5.13 shows the vehicle state distribution over time for the two scenarios M-MANH-Wed and M-MANH-Sun with DIS+LS. Although the demand patterns are substantially different, the behavior of our algorithm is similar. Trip requests are mainly rejected at peak demand times with some rejections throughout the day caused by remote trip requests with no nearby vehicles.

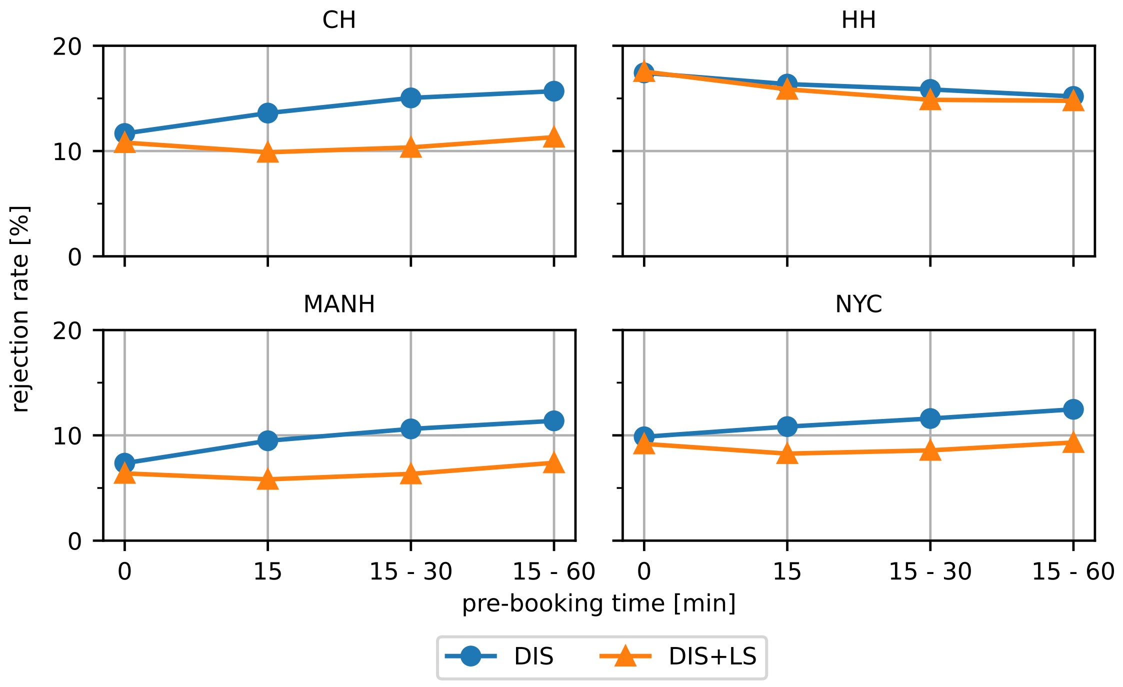


Figure 5.12: Rejection rates for different pre-booking durations.

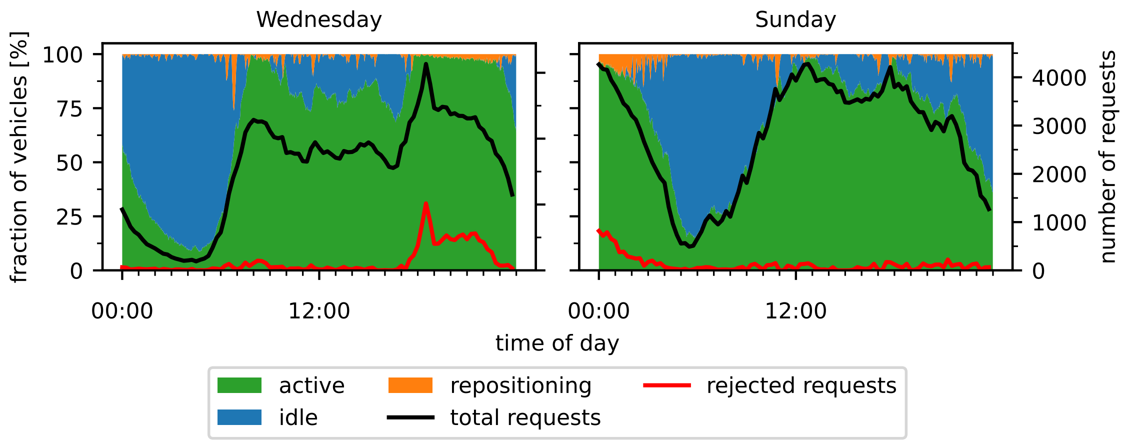


Figure 5.13: Vehicle states on Wednesday and Sunday for instances M-MANH-Wed and M-MANH-Sun with DIS+LS.

5.6.2.11 Local Search Time Limit

We evaluate two settings regarding the local search time limit. One with a limit of 200 ms. This setting emulates a real-time usage of the algorithm in which the available computation time between two trip requests may be as small as 200 ms during demand peaks. Second, we also consider a setting with unlimited computational time for our local search. The idea is that this gives us the ability to see the potential of the local search if computational time was no issue. Such results could be obtained by a more efficient implementation of

Table 5.18: Results for different local search time limits.

Data	Time Limit	Rej [%]	Wait [s]	Ride [s]	TTv [min]	TTvr [s]	RT [min]
CH	200	10.80	200.52	**593.13**	929.69	350.00	70.51
	inf	**10.77**	**200.48**	593.16	**929.52**	**349.81**	69.58
HH	200	17.55	174.95	476.19	842.71	456.72	0.73
	inf	17.55	174.95	476.19	842.71	456.72	0.72
MANH	200	6.39	222.96	293.07	1028.68	156.16	83.18
	inf	6.39	222.96	293.07	1028.68	156.16	82.94
NYC	200	**9.19**	216.69	**375.70**	**1037.41**	**217.17**	151.07
	inf	9.28	**216.51**	375.87	1037.71	217.45	150.97
ALL	200	**10.98**	203.78	**434.52**	**959.62**	**295.01**	76.37
	inf	11.00	**203.73**	434.57	959.66	295.04	76.05

the local search or more powerful computational resources. The results are summarized in Table 5.18. As we may see, for two datasets (HH and MANH), the computational results are identical (aside from some minor running time fluctuations). The local search is able to exhaustively search the search space within the time limit. For the other two datasets, there are minor differences. However, the running times and performance metrics are still very similar. Hence, for our default scenario, the increased local search time limit does not yield any significant improvements. In scenarios with a larger search space (e.g. with larger time windows), this behavior may be different. However, we conclude that additional computational power would not be the most promising way to improve the local search. Instead, one should look towards more complex search operators or a more refined guidance for the search.

5.7 Conclusions

In this chapter, we introduced a vehicle routing approach for the VRPDRS consisting of a fast dispatching algorithm and a local search improvement phase. Through extensive computational studies, we showed that our approach is capable of handling a diverse range of scenarios in real-time. The local search manages to improve the rejection rates and efficiency of vehicle routes. It is particularly important in scenarios with long time windows, high vehicle occupation or pre-booked trip requests. In such settings, it may no longer be sufficient to employ a purely dynamic dispatching algorithm. Our results show that local search techniques for vehicle routing can be integrated into a highly dynamic planning process. Combined with a dispatching approach, this is a promising solution that offers both immediate response times for customers and good solution quality. Regarding the impact of ride-sharing services on city-wide traffic, our evaluations indicate that it is possible to significantly reduce the traffic load compared to ride-hailing or taxi services. And even compared to the usage of personal vehicles, the overall vehicle travel time may be reduced. However, for the latter to be achieved, a high demand density is necessary.

The usage of established local search techniques for vehicle routing problems in a highly dynamic setting is a promising direction for future research. While we have shown that simple search operators can be integrated into a dynamic planning process, we see the potential to utilize more complex neighborhoods and metaheuristics in this context. In addition, the dispatching and local search algorithms could be extended to explicitly consider pre-booked trip requests. As our results show, the integration of these requests into dynamic vehicle routing algorithms can lead to adverse effects. Hence, we envision a combination of static vehicle routing approaches that process pre-booked requests with dynamic approaches that build upon an already existing plan. We believe that it is highly desirable for dynamic ride-sharing services to be able to accommodate reservations for trip requests in advance and use this information to build more efficient routing plans. In general, we see the usage of stochastic information on future trip request demand, vehicle utilization, and traffic congestion as one major research direction which we will partly address in the next chapter.

6 Idle Vehicle Repositioning for Dynamic Ride-Sharing

In the previous chapter, we presented a vehicle routing algorithm for the VRPDRS. As illustrated by the computational results, the lack of a repositioning mechanism for idle vehicles may lead to a serious deterioration in the system's performance. Hence, in this chapter, we focus on the inclusion of a forecast-based repositioning algorithm that anticipates future trip requests and repositions idle vehicles accordingly. This chapter is based on the following two articles:

> M. Pouls et al. (2020). Idle Vehicle Repositioning for Dynamic Ride-Sharing. *Computational Logistics.* Ed. by E. Lalla-Ruiz et al. Vol. 12433. Cham: Springer International Publishing, pp. 507–521. DOI: `10.1007/978-3-030-59747-4_33`.
>
> M. Pouls et al. (2022). Adaptive forecast-driven repositioning for dynamic ride-sharing. *Annals of Operations Research.* DOI: `10.1007/s10479-022-04560-3`

6.1 The Idle Vehicle Repositioning Problem for Dynamic Ride-Sharing

In the previous chapter, we have focused purely on the VRPDRS, i.e. the assignment of incoming trip requests to vehicles. However, as our computational results in Section 5.6.2.3 have illustrated, the lack of an anticipatory component that predicts future trip requests and repositions vehicles accordingly severely impacts the overall system performance. Figure 6.1 illustrates this aspect by comparing vehicle positions after several hours of service in simulation studies without and with repositioning. Without a repositioning mechanism, vehicles become stuck in low-demand areas. In turn, trip requests in other areas are rejected due to the lack of nearby vehicles as the vehicles in more distant regions cannot reach the trip requests within their pickup time windows. This phenomenon may be avoided through the usage of a repositioning mechanism that actively steers idle vehicles toward high-demand regions.

In practical applications with self-employed drivers, i.e. services like Uber and Lyft, this problem is tackled by incentivizing drivers to reposition towards areas with a low vehicle supply. For instance, Uber employs a mechanism called “surge-pricing" that raises prices in areas with excess demand and thereby increasing revenue opportunities for drivers (Uber 2020). Lyft utilizes a similar mechanism named “personal power zones" (Ong et

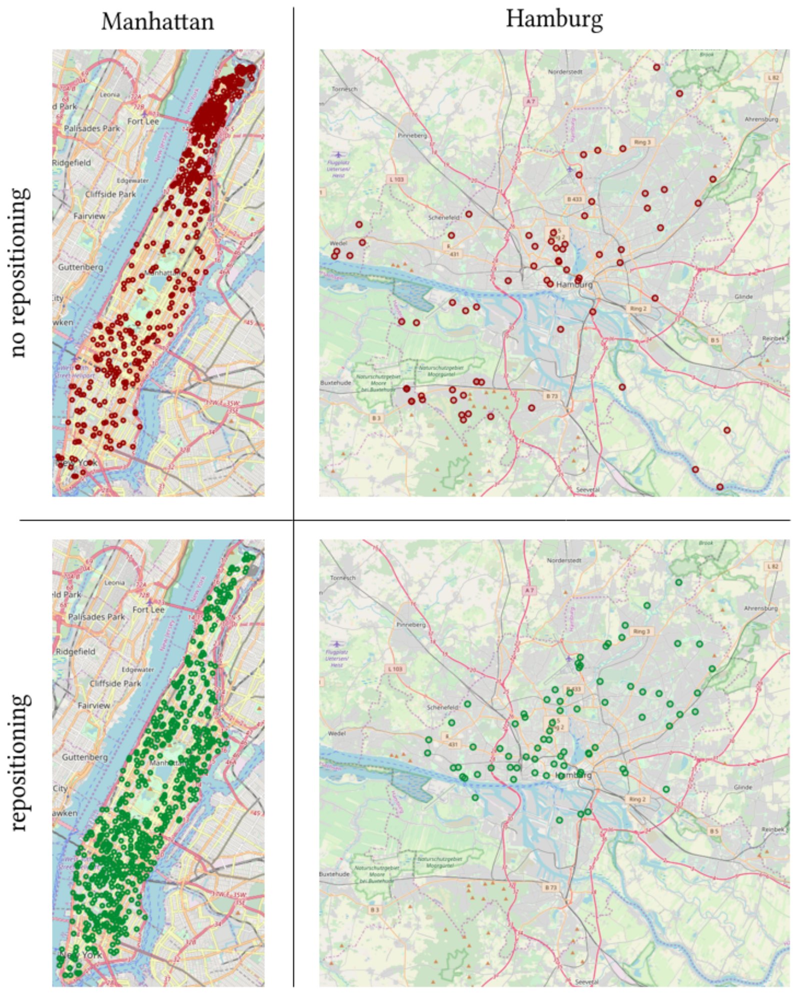


Figure 6.1: Dots show vehicle positions without (red) and with repositioning (green) after several hours of service for simulation scenarios in Manhattan and Hamburg.

al. 2021), which similarly incentivizes drivers monetarily to move towards zones with lacking vehicle supply. However, in use cases with a central fleet operator, we believe that a central repositioning strategy can be beneficial for the overall system performance compared to such decentral repositioning mechanisms. Existing approaches from literature on central repositioning in the context of dynamic ride-sharing may be roughly divided into two groups. Firstly, explicit repositioning algorithms that treat repositioning as a separate decision from vehicle routing. Secondly, anticipatory routing algorithms that integrate vehicle routing and repositioning. In the remainder of this chapter, we propose a repositioning algorithm that falls into the first group. The main objective of our approach is to increase the number of trip requests served in a ride-sharing system. Our major contributions are threefold:

- We propose a forecast-based repositioning algorithm that is applicable in real-time to large-scale dynamic ride-sharing systems. Our approach is modular and may be easily combined with existing vehicle routing solutions. For the remainder of this work, we study repositioning in combination with our vehicle routing algorithm presented in Chapter 5.

- We introduce an integrated adaptive parameter tuning technique to adequately consider temporal and spatial differences in the number of trip requests that vehicles are expected to serve. This is crucial in the context of ride-sharing services and ensures that our approach can be applied to new settings without requiring extensive prior configuration or parameter tuning. In addition, the algorithm automatically adapts to changes in the relevant system parameters such as the overall demand.
- We perform an extensive computational evaluation on a diverse set of scenarios derived from four real-world datasets and compare our algorithm to a myopic benchmark algorithm. Our results show that the forecast-based approach is able to reduce the number of rejected trip requests and additionally also improves customer waiting times.

The remainder of this chapter is structured as follows. Section 6.2 summarizes the related work in the fields of repositioning, anticipatory vehicle routing, and demand forecasting. In Section 6.3 we detail our own repositioning algorithm. Subsequently, Section 6.4 presents our computational evaluations. Finally, Section 6.5 summarizes our findings and presents some directions for potential future research.

6.2 Related Approaches in Repositioning and Demand Forecasting

To the best of our knowledge, there are relatively few papers dealing with repositioning explicitly in the context of large-scale dynamic ride-sharing applications. However, there exist several closely related fields. There is a large body of work on the dynamic vehicle routing problem with stochastic customers (DVRPSC). In this vehicle routing variant, part of the trip requests arrive dynamically and there is some form of stochastic knowledge about trip request arrivals that may be exploited during planning. There also exists a large number of works regarding repositioning in other mobility-on-demand services besides ride-sharing. For instance, repositioning is a widely considered problem in car-sharing systems. In these systems, customers rent a vehicle for a desired time period. Repositioning is particularly important for one-way systems in which customers may drop off the vehicle at a different location than their pickup. However, there are some key differences between car-sharing and dynamic ride-sharing. In car-sharing, there are no dedicated drivers and customers drive the rented vehicles themselves. In addition, there is no sharing between different customers, a vehicle is assigned to exactly one customer at a time.

We structure our literature review as follows. We first consider related work regarding repositioning in dynamic ride-sharing systems. These are closely related to the algorithm presented in this chapter. Subsequently, we review literature concerning the DVRPSC as well as repositioning approaches for car-sharing and taxi services. Finally, we discuss existing approaches for short-term travel demand forecasting, a vital component in our repositioning approach.

6.2.1 Repositioning in Dynamic Ride-Sharing

Most closely related to the algorithm presented in this chapter are works dealing with repositioning in the context of dynamic ride-sharing systems. A summary of existing approaches is provided in Table 6.1 with the following criteria.

Routing interaction We differentiate between approaches where repositioning is explicitly performed by a *separate* algorithm versus algorithms that *integrate* some form of anticipation of future trip requests into the vehicle routing algorithm.

Objective Most approaches strive to maximize the number of served trip requests while secondarily minimizing vehicle travel times. Sometimes this secondary objective is not explicitly formulated but rather a side-effect as maximizing the served trip requests often correlates with planning efficient routes. One paper uses a different objective function as the rejection of trip requests is not allowed.

Solution approach We give a short description of the utilized solution methodology. Similar to the papers discussed in Section 5.2.1, there is a group of works that all build upon the graph-based solution approach by Alonso-Mora et al. (2017a).

Real-world datasets, trip requests, and vehicle fleets As before in Section 5.2.1, we provide information concerning used real-world datasets, numbers of trip requests, and the configuration of the vehicle fleet. For a more detailed discussion of these aspects please refer to Section 5.2.1.

In the following we discuss the papers summarized in Table 6.1 in detail. Alonso-Mora et al. (2017a) propose a reactive repositioning policy with the idea of sending idle vehicles to the desired pickup locations of rejected trip requests. Given a batch of rejected requests, idle vehicles are matched to the corresponding pickup locations while minimizing travel times for repositioning movements. We use a similar approach as a benchmark for our solution algorithm. In a follow-up paper, Alonso-Mora et al. (2017b) present a more refined approach similar to the sampling-based algorithms for the DVRPSC presented in the following section. They include predicted trip requests in their vehicle routing algorithm. These requests are served with a lower priority than actual trip requests. The authors show that this approach leads to reduced waiting times and in-car travel delays compared to the reactive repositioning from their previous work (Alonso-Mora et al. 2017a). However, no noticeable improvement in the number of rejected trip requests is achieved.

Paper	Routing interaction	Objective	Solution approach	Real-world datasets	Trip requests	Fleet size	Vehicle capacity
Alonso-Mora et al. (2017a)	separate	max. requests served and min. vehicle travel time	reactive	NYC	460,700 / day	1,000 – 3,000	1 – 10
Alonso-Mora et al. (2017b)	integrated	max. requests served and min. vehicle travel time	graph-based matching + request sampling	NYC	460,700 / day	1,000 – 3,000	2 – 4
Chow and Jung (2019)	separate	max. requests served	depot or zone-based repositioning policies	NYC	145,643 / 6 hours	3,000 – 7,000	6
Riley et al. (2020)	separate	min. waiting time and min. vehicle travel time	MIP-based zone rebalancing + demand forecast	NYC	59,820 / 2 hours	2,000	4
Shah et al. (2020)	integrated	max. requests served and min. vehicle travel time	graph-based matching + RL & ADP	NYC	19,820 / hour	1,000 – 3,000	2 – 10
Lowalekar et al. (2021)	integrated	max. requests served and min. vehicle travel time	graph-based matching + zone clustering + benders decomposition	NYC	403,770 / day	1,000 – 10,000	1 – 10
Pouls et al. (2020) Pouls et al. (2022) this work	separate	max. requests served and min. vehicle travel time and min. repositioning movements	MIP-based repositioning + demand forecast	NYC, Hamburg, Chengdu	429,855 / day	72 – 1,512	2 – 6

Table 6.1: Related work on repositioning for dynamic ride-sharing.

Chow and Jung (2019) present two policies in which vehicles reposition according to historical pickup probabilities. Vehicles either move to a zone or a depot. The probability of selecting a zone or depot is proportional to the historical distribution of trip requests. The authors compare these approaches to a setting without repositioning and show that both repositioning policies improve the request acceptance rate at the cost of an increase in the total distance traveled by vehicles. In contrast to our work, the authors do not consider detailed information about supply and demand. In particular, neither the current configuration of the vehicle fleet nor the total demand is considered during repositioning.

Riley et al. (2020) propose an approach based on a mixed-integer programming (MIP) formulation similar to the one presented in this work. They use a two-step formulation that first determines the number of vehicles to be repositioned between a pair of zones and subsequently selects specific vehicles. Their algorithm integrates a demand forecast containing the number of trip requests between each pair of zones. Hence, in contrast to this work, they expect a more fine-grained forecast which may not be available in practice. Moreover, they do not allow for the rejection of trip requests but rather penalize long customer waiting times. They evaluate their approach on data from New York City and show that their repositioning approach decreases customer waiting times compared to performing no repositioning.

Shah et al. (2020) propose a learning-based approach that assesses the future value of routing decisions. However, the approach does not explicitly reposition idle vehicles. It rather considers the future value of routing decisions when assigning trip requests to vehicles. They evaluate their algorithm on data from New York City and show that it yields a decrease in rejected trip requests compared to myopic routing approaches.

Lowalekar et al. (2021) propose another approach that includes samples of future trip requests to build routes that are suitable to accommodate upcoming requests. The authors evaluate their approach on two real-world datasets from NYC and another undisclosed location and show that the trip request rejection rate is reduced compared to myopic approaches.

6.2.2 Dynamic Vehicle Routing with Stochastic Customers

In the DVRPSC, part of the trip requests arrive dynamically and there is some form of exploitable stochastic information about these trip requests. The problem has been widely studied in literature. A selection of solution approaches is summarized in Table 6.2. For more extensive literature reviews we refer the reader to Ritzinger et al. (2016) and Ulmer et al. (2020). As before, we report the solution approaches and largest instance sizes per paper. In addition, we adopt the classification by Ulmer et al. (2020) that distinguishes between the following general solution methods.

Lookahead algorithms (LAs) Approaches that use a lookahead, for instance, samples of anticipated trip requests, to improve routing decisions.

Policy function approximation (PFA) Algorithms that aim to approximate a policy and are often inspired by decision-making in practice.

Value function approximation (VFA) Procedures that derive a value function for routing decisions via simulations and related learning techniques.

Among the earliest works in the field of the DVRPSC are waiting strategies (Mitrović-Minić and Laporte 2004; Ichoua et al. 2006; Thomas 2007), which represent examples of PFA algorithms. In these algorithms, the aim is to decide when and where a vehicle should wait while executing a route in order to be well-positioned for dynamically arriving trip requests. While these approaches yield benefits for many application settings such as parcel delivery (Mitrović-Minić and Laporte 2004), the problems arising from these applications are structurally different from the setting of ride-sharing. For instance, they tend to be less dynamic as a large portion of customer requests is known in advance, while in this work we consider purely dynamic trip requests. Additionally, vehicles in ride-sharing tend to have little to no waiting times while executing a route as dynamically arriving customers

Table 6.2: Related work on the DVRPSC.

Paper	Solution approach	Classification by Ulmer et al. (2020)	Instance size
Bent and Van Hentenryck (2004)	MSA	LA	100 / day
Mitrović-Minić and Laporte (2004)	Waiting strategy	PFA	1,000 / 10 hours
Ichoua et al. (2006)	Waiting strategy	PFA	36 / hour
Bent and Van Hentenryck (2007)	MSA	LA	100 / day
Thomas (2007)	Waiting strategy	PFA	50 / day
Mes et al. (2010)	Auction	VFA	4.5 / hour
Schmid (2012)	ADP	VFA	90 / day
Ferrucci et al. (2013)	TS with sampling	LA	150 / day
Ulmer et al. (2018)	ADP	VFA	100 / 6 hours
Ulmer et al. (2019)	ADP	VFA	100 / 6 hours
Voccia et al. (2019)	ADP	VFA	192 / day

want to be serviced immediately. Moreover, when transporting passengers, it is difficult to justify waiting times whenever there are customers aboard the vehicle or waiting for its arrival. In contrast, for applications such as parcel logistics, this aspect is less of a problem. Therefore, in our view, such waiting strategies are not directly applicable to the use case considered in this paper.

The most prevalent approach for lookahead algorithms is the inclusion of samples of predicted customers in the routing algorithm. Among these sampling-based algorithms are the multiple scenario approach (MSA) by Bent and Van Hentenryck (2004) and Bent and Van Hentenryck (2007), in which multiple routing plans are generated based on different samples of future customers and a so-called “distinguished" plan is selected via consensus mechanisms. Ferrucci et al. (2013) also propose a sampling-based approach in the form of a tabu search (TS) that includes sampled future customers. Due to their computational complexity, most sampling-based approaches have not been tested on large instance sizes as commonly seen in dynamic ride-sharing.

A third major direction of DVRPSC research are VFA algorithms such as approximate dynamic programming (Mes et al. 2010; Schmid 2012; Ulmer et al. 2018; Ulmer et al. 2019; Voccia et al. 2019). The idea behind these approaches is to determine a value function for routing decisions that incorporates their impact on the handling of future trip requests. From the practical perspective of ride-sharing applications, these algorithms have two main drawbacks. First, they need a relatively large amount of training data to approximate the value function. Hence, launching a ride-sharing service in a region where no prior data is available becomes a challenge. Second, these VFA approaches are also not tested on large instances and may not deliver the necessary computational performance for processing large numbers of trip requests.

6.2.3 Repositioning in Car-Sharing, Taxi Operations, and Other Mobility-as-a-Service Applications

Vehicle repositioning is a widely studied problem in the context of car-sharing services. One may differentiate between operator-based and user-based repositioning. In the former, the car-sharing operator employs personnel to reposition vehicles while in the latter car-sharing users are incentivized to pick up or drop off cars at certain locations. In our literature review, we focus on operator-based repositioning as it shares a closer resemblance with central repositioning approaches for ride-sharing. A summary of recent works in this field is presented in Table 6.3. A more detailed review may be found in Huang et al. (2020b).

Algorithms are often based on MIP formulations (Nourinejad and Roorda 2014; Repoux et al. 2015; Boyacı et al. 2017; Gambella et al. 2018; Xu and Meng 2019; Huang et al. 2020b), but also include metaheuristics (Bruglieri et al. 2019) and Markov chain based models (Repoux et al. 2019). There are some key differences compared to the application setting of dynamic ride-sharing. Most importantly, in car-sharing systems, there is a one-to-one relation between a customer and a vehicle. In contrast, ride-sharing allows for multiple customers

Table 6.3: Related work on repositioning for car-sharing.

Paper	**Solution approach**	**Instance size**
Nourinejad and Roorda (2014)	MIP	200 / day
Repoux et al. (2015)	MIP	200 / day
Boyacı et al. (2017)	MIP	300 / day
Gambella et al. (2018)	MIP	50 / day
Bruglieri et al. (2019)	ALNS	100
Repoux et al. (2019)	Markov chain model	400 / day
Xu and Meng (2019)	MIP	125 / day
Huang et al. (2020b)	MIP	125,000 / day

to share the same vehicle. Therefore, one key challenge when making repositioning decisions in a ride-sharing system is to estimate how many vehicles are needed to serve a given number of trip requests. As this depends on the structure of the trip requests, it may vary between datasets but also between locations or the time of day within the same dataset. On the other hand, repositioning approaches for car-sharing systems focus on specific aspects of that application domain. These include scheduling the personnel that carries out the repositioning movements, considering different operation modes (two-way, one-way, free-floating) and different reservation schemes. Hence, we believe that due to the structural differences between car- and ride-sharing, specific solution approaches are needed for both applications.

Repositioning approaches have also been proposed for ride-hailing and classic taxi services. In these services, no sharing takes place and only one group of customers uses the vehicle at a time. Both Li et al. (2011) and Powell et al. (2011) use GPS traces of taxis to identify profitable regions. In contrast to most other works presented in this section, these approaches take the viewpoint of the taxi driver and aim to optimize the profit, while our goal is to optimize the system-wide performance. Additionally, they do not take into account the vehicle-sharing aspect and assume that a vehicle serves at most one customer at a time. Syed et al. (2021) propose a repositioning algorithm for ride-hailing services that uses similar concepts as presented in this work. They reposition vehicles between regions based on the estimated imbalance between supply and demand. The authors evaluate the approach on a simulated ride-hailing service based on New York City taxi data. However, their solution method could also be adapted for other MOD services such as dynamic ride-sharing.

6.2.4 Short-Term Travel Demand Forecasting

In our solution approach, we utilize a short-term forecast of the anticipated trip requests. The general field of short-term travel demand forecasting and related spatio-temporal prediction problems has been studied extensively. For reviews, we refer the reader to Vlahogianni et al. (2004) and Vlahogianni et al. (2014). Most approaches provide information in an aggregated form, i.e. they offer a forecast of the total number of trip requests for a given area and interval of time rather than predicting individual trip requests (Vlahogianni et al. 2014). Classical approaches include time series models such as ARIMA or Kalman filters (Li et al. 2012; Lippi et al. 2013) as well as statistical learning (Huang et al. 2020a). More recently, convolutional neural network (CNN) and long short-term memory (LSTM) recurrent neural networks have emerged as suitable techniques for modeling the complex spatial and temporal dependencies typically found in these forecasting problems (Zhang et al. 2018; Liao et al. 2018; Yao et al. 2018; Ke et al. 2017; Yao et al. 2019). For instance, Yao et al. (2018) and Yao et al. (2019) combine a CNN for modeling spatial dependencies with a LSTM architecture that reflects the temporal aspects. They evaluate their approach on New York City taxi data and achieve a significant improvement over a simple historical average as well classic machine learning algorithms such as gradient boosting.

In the remainder of this work, we will not focus on the forecasting methodology itself but rather on the usage of a forecast to improve planning results. However, the assumptions of our algorithm regarding the structure of the forecast conform to state-of-the-art forecasting techniques, i.e. they expect the predicted number of trip requests per area and interval of time as an input. This way, we ensure that these approaches can be used in combination with our algorithm.

6.2.5 Contribution

Throughout the previous sections, we have presented a summary of related solution approaches in the field of idle vehicle repositioning. We believe that there are still some research gaps, particularly concerning the consideration of the current system state and anticipated demand as well as the integration with existing vehicle routing approaches. We aim to address these aspects in our work. Compared to the existing literature, the following novel aspects are considered in this work.

Modular and adaptive repositioning algorithm In our algorithmic design, we strive to consider several practically relevant aspects. Firstly, the repositioning algorithm should be applicable to large-scale scenarios with several hundreds of thousands of trip requests per day. Secondly, the algorithm should be modular and easily integrated with existing vehicle routing solutions. Lastly, the algorithm should be easily applied to new application settings, for instance, a new city in which the ride-sharing service is introduced. Hence, it should adapt to new environments well without the necessity for training data or a priori parameter tuning.

Integration of forecasting techniques Our repositioning algorithm works in conjunction with existing forecasting models. To ensure this, we consider the state-of-the-art techniques in the realm of short-term travel demand forecasting and ensure that the input data structure of our algorithm conforms to the output that these models commonly produce. Hence, if a service provider already has existing forecasting models, these could be used in combination with our repositioning algorithm.

6.3 Forecast-Driven Repositioning

In this section, we describe our forecast-driven repositioning algorithm (FDR). We first present a brief overview of the general ideas behind our algorithm. Subsequently, we consider the envisioned planning process in detail and present the separate phases of our algorithm.

6.3.1 Problem Statement and Algorithm Overview

The main objective in idle vehicle repositioning for dynamic ride-sharing is to maximize the number of served trip requests by repositioning vehicles and enabling them to reach trip requests on time that would otherwise have to be rejected. A secondary objective or side-effect of this approach is the minimization of lead times when approaching a new trip request. This leads to improved fleet utilization as well as reduced waiting times for customers.

The central component of our approach is a MIP model (FDR-M) that is solved in regular intervals. In this model, we aim to balance supply and demand by maximizing the sum of covered demand and minimizing the number of repositioning movements as well as vehicle travel times. FDR-M work on a spatially aggregated level. Hence, we assume that our region under study is partitioned into a discrete set of areas. Anticipated demand is given by a forecast that outputs the expected number of trip requests originating in each area over a forecast horizon. Supply on the other hand is provided by the vehicle fleet. We assume that vehicles may cover demand in the neighborhood of their current location. Alternatively, if a vehicle is selected for repositioning, we assume that it will cover demand in the neighborhood of its assigned repositioning target. This neighborhood is intuitively defined as the set of areas that may be reached within the maximum allowed waiting time of a customer. A single vehicle is assumed to cover multiple trip requests over the forecast horizon. However, the precise number varies drastically by dataset, location, and time of day. For instance, at night, when demand tends to be lower, vehicles serve fewer customers in the same amount of time as fewer trip requests can be combined into a single-vehicle route. Similar behavior may be observed when comparing low- and high-demand areas. Vehicles in high-demand areas serve more customers within the same time period. Thus, we integrate an adaptive parameter tuning technique into our approach that estimates the expected number of served trips for a vehicle located in a certain area. As this parameter is updated each time we solve FDR-M, it reflects the spatial and temporal differences in the utilization of vehicles. Based on this adaptive parameter tuning and the current vehicle schedules, we may calculate the supply provided by our vehicle fleet. By repositioning idle vehicles, we are now able to shift supply from low-demand areas to areas with excess demand. Repositioning decisions in FDR-M are taken on an aggregated level. As an output, the model determines the number of vehicles repositioned between each pair of areas. Translating these decisions into actionable repositioning assignments is part of a rolling-horizon planning process that we explain in the following section. Subsequently, we introduce FDR-M itself alongside the necessary notation and the adaptive parameter tuning technique.

6.3.2 Planning Process

FDR-M is embedded into a rolling horizon planning process which is triggered at a regular interval I. The four main steps of the planning process are as follows:

1. Obtain an up-to-date demand forecast from an external forecasting module.
2. Perform the adaptive parameter calculation as detailed in Section 6.3.4.
3. Solve FDR-M as given in Section 6.3.5.
4. Determine an optimal assignment of vehicles to repositioning targets (Section 6.3.6).

In the first step, a demand forecast is obtained that yields the number of expected trip requests for a set of areas within a forecast horizon h. Combined with the current state of the vehicle fleet, this forecast serves as an input for FDR-M. In the second step, parameters for FDR-M are adaptively determined based on the current performance of the vehicle fleet. This enables us to adjust to varying spatial and temporal conditions. Subsequently, we solve FDR-M which determines repositioning assignments on an aggregated level. As an output, we obtain the number of vehicles repositioned between any pair of areas i and j, denoted as $x_{i,j}$. Lastly, these aggregated values are translated into actionable decisions. For this purpose, we sample target locations from each target area j and subsequently assign idle vehicles to these locations in a way that minimizes the overall travel times for repositioning. In the remainder of this section, we will take a detailed look at steps $2-4$ of the planning process which constitute the core portion of our algorithm. Step 1 is not discussed in detail as forecasting models are not the focus of this work. For a brief review of relevant forecasting methods, we refer the reader to Section 6.2.4.

6.3.3 Notation Overview

The relevant notation used throughout this section is summarized in Table 6.4. K denotes the set of all vehicles. This set may be further subdivided into idle vehicles K^{id}, active vehicles currently serving trip requests K^{act} and repositioning vehicles K^{re}. As mentioned previously, we assume a partitioning of the region under study into discrete areas A. For the remainder of this work, we utilize a partitioning into square grid cells similar to Chapter 5. However, note that the utilized grid is different from the one used in the vehicle routing algorithm. In particular, it tends to be beneficial to utilize a less fine-grained grid partitioning, as a certain level of aggregation is desirable. The size of the grid cells used

Table 6.4: Notation for forecast-driven repositioning.

Sets	
A	Areas
A^{re}	Valid target areas for repositioning
K	Vehicles
$K^{id}\|K^{act}\|K^{re}$	Idle | active | repositioning vehicles
$K_i^{id}\|K_i^{act}\|K_i^{re}$	Idle | active | repositioning vehicles per area $i \in A$
K_i^N	Vehicles in the neighborhood of i at the start of h^-
RL_i	Feasible repositioning target locations in $i \in A$
N_i	Neighborhood of i
T^{re}	Repositioning targets sampled from RL_i, $i \in A$
Parameters	
α_k	Active percentage of vehicle k
$\hat{d}_i$	Demand forecast for $i \in A$
$\hat{D}$	Cumulated demand forecast for all areas $i \in A$
d_k^-	Number of performed delivery operations by vehicle k in h^-
d_k^+	Number of planned delivery operations by vehicle k in h^+
$\hat{rs}_i$	Expected number of requests served by a vehicle in $i \in A$ in h^+
$\bar{rs}_k$	Potential number of requests served by $k \in K$ in h^-
I	Interval of time in which algorithm FDR is run
g_{re}^{size}	Side length grid cells for repositioning
h	Forecast horizon in minutes
$h^+\|h^-$	Time periods covering the previous and next h minutes
k_N^{min}	Minimum number of vehicles for neighborhood calculations
p_k^-	Performed pickup operations by vehicle k in h^-
p_k^+	Planned pickup operations by vehicle k in h^+
S_i^{act}	Supply provided by active vehicles in area $i \in A$
S_i^{re}	Supply provided by repositioning vehicles to area $i \in A$
$tt_{i,j}$	Travel time from area i to j
tt_t^k	Travel time of vehicle k to location t
tt^{max}	Maximum travel time $\max_{i,j \in A} tt_{i,j}$
w^{cov}	Objective function weight for covered demand
w^{re}	Objective function weight for repositioning movements
$w^{tt,cov}$	Objective function weight for coverage travel times
w_i	Objective function weight for coverage of demand in area $i \in A$
Decision variables	
$c_{i,j} \in \mathbb{R}_0^+$	Provided demand coverage from area i to j
$x_{i,j} \in \mathbb{N}_0^+$	Number of vehicles repositioned from area i to j
$a_{k,t} \in \{0, 1\}$	Assignment of vehicle k to repositioning target t

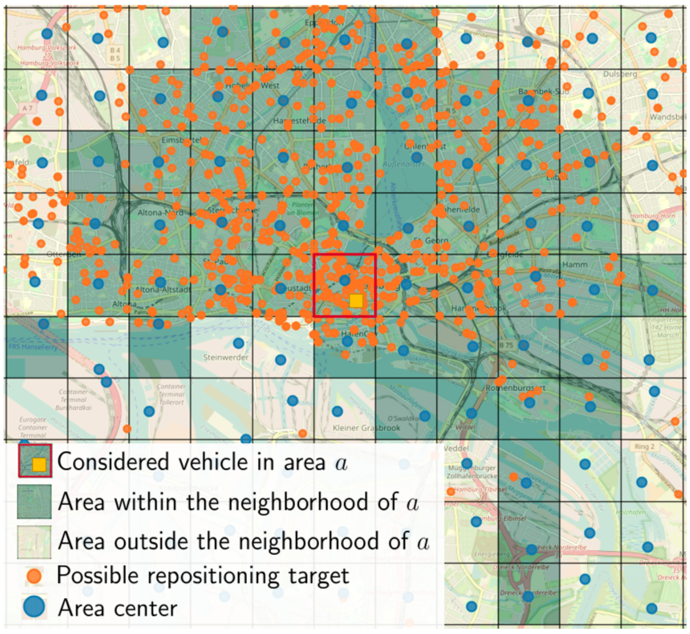

Considered vehicle in area a
Area within the neighborhood of a
Area outside the neighborhood of a
Possible repositioning target
Area center

Figure 6.2: Illustration of the grid-based map partitioning, the neighborhood of the considered area a, the area centers, and possible repositioning targets.

for repositioning is given by g_{re}^{size} and denotes the length of the sides. Figure 6.2 illustrates the utilized grid among other key concepts that will be explained throughout this section. For the purpose of travel time calculations, we assume that areas are represented by their center and calculate a travel time $tt_{i,j}$ between the centers of two areas $i, j \in A$. Centers are not the exact centroid of an area but rather the closest node on the road network. Consequently, travel times are calculated from the shortest paths on the road network. The maximum travel time between any pair of areas is denoted as $tt^{max} = \max_{i,j \in A} tt_{i,j}$. We may now further divide the sets of vehicles by area $i \in A$ as K_i^{id}, K_i^{act}, K_i^{re}. Note that the sets K_i^{id} and K_i^{act} contain those vehicles currently situated in area i. K_i^{re} on the other hand consists of vehicles currently repositioning towards i. Vehicles may only be repositioned to valid target areas $A^{re} \subseteq A$. In this study, we limit A^{re} to areas with at least one prior pickup. This is necessary as we sample specific repositioning target locations RL_i for each $i \in A^{re}$ from past pickup locations. An example of possible repositioning targets may be seen in Figure 6.2. In practical applications, A^{re} might be determined based on suitable parking spots for vehicles. Our model works on a demand forecast denoted as $\hat{d}_i$ for each $i \in A$. This forecast gives us the expected number of trip requests originating in an area i within a forecast horizon of h minutes. h^+ and h^- denote time periods covering the previous and next h minutes respectively.

6.3.4 Adaptive Parameter Calculation

After obtaining a demand forecast, the first main step of our algorithm is the adaptive calculation of relevant parameters. More specifically, we aim to estimate the number of trip requests that a vehicle situated in area $i \in A$ is expected to serve within h^+. We denote this expected number of requests served as $\hat{rs}_i$. This value may vary drastically between different scenarios, but also within the same scenario depending on the time of day and considered area. For instance, vehicles tend to be able to serve more trip requests in the same time frame in high-demand areas as it is possible to build more efficient vehicle routes.

In order to consider these aspects, we consider the current performance of our vehicle fleet and determine the potential number of trip requests that each vehicle could have served throughout the last h^- minutes as in Equation 6.1.

$$\bar{rs}_k = u^{tar} \cdot \frac{1}{\alpha_k} \cdot \frac{p_k^- + d_k^-}{2} \qquad k \in K \quad (6.1)$$

p_k^- and d_k^- denote the number of pickup and delivery operations performed by k in h^-. Thus, dividing the sum of these values by two in the right-hand side of the equation gives us the number of trip requests served in h^-. However, k may have been idle for a portion of h^-, therefore this number does not reflect the potential number of served requests at full utilization. To account for this fact we multiply by a factor of $\frac{1}{\alpha_k}$ where α_k is the percentage of time that k was active in h^-. This gives us an upper bound on the number of trip requests that the vehicle could have realistically served. In general, a 100 % usage rate of vehicles is not achievable in practical scenarios. Hence, we multiply by a target utilization rate u^{tar}, which may be determined empirically. Based on preliminary studies, we estimate that an average utilization rate of 90 % ($u^{tar} = 0.9$) is a realistic target.

Based on the current vehicle performance, we may now estimate the expected number of served trip requests $\hat{rs}_i$ for each area i as defined in Equation 6.2. This value yields an estimation of how many trip requests a vehicle starting in area i will serve in the time period h^+.

$$\hat{rs}_i = \frac{\sum_{k \in K_i^N} \bar{rs}_k}{|K_i^N|} \qquad i \in A \quad (6.2)$$

To calculate $\hat{rs}_i$, we consider all vehicles K_i^N that were located in the neighborhood N_i of i at the start of the previous horizon h^-. This neighborhood is defined as the set of areas $j \in A$ that may be reached from i within the maximum waiting time w_r of a request. Thus, N_i is defined as $N_i = \{j \in A | tt_{i,j} \leq w_r\}$. In our studies, we assume that all trip requests have the same w_r. In cases where this is not applicable, an average value could be used. To achieve a reliable estimate, we furthermore ensure that K_i^N contains at least k_N^{min} vehicles. If this is not the case, the neighborhood N_i is grown iteratively by including the next closest area until k_N^{min} vehicles are reached. We may then calculate $\hat{rs}_i$ as the average of $\bar{rs}_k$ for all vehicles in K_i^N.

Based on $\hat{rs}_i$, we are now able to estimate our available supply in any area. Supply provided by repositioning vehicles, i.e. vehicles that have been selected for repositioning in previous runs of the repositioning algorithms, is calculated as defined in Equation 6.3. We assume that each vehicle repositioning to area i will serve approximately $\hat{rs}_i$ requests over the coming forecast horizon h^+. Thus, we merely multiply the number of repositioning vehicles by $\hat{rs}_i$.

$$S_i^{re} = |K^{re}| \cdot \hat{rs}_i \qquad i \in A \quad (6.3)$$

Supply provided by active vehicle is calculated in a similar fashion in Equation 6.4. However, in this case, we need to consider the current vehicle schedules. Thus, for each vehicle, we adjust the number of expected served trip requests based on the current planned pickup (p_k^+) and delivery (d_k^+) operations.

$$S_i^{act} = \sum_{k \in K_i^{act}} \hat{rs}_i - \frac{p_k^+ + d_k^+}{2} \qquad a \in A \quad (6.4)$$

Finally, supply is also provided by idle vehicles that are either selected to reposition to another location in the subsequent period or that stay at their current position. This is expressed through decisions in our repositioning model that is presented in the next section.

6.3.5 Mathematical Model for Forecast-Driven Repositioning

The complete repositioning model FDR-M is given in equations (6.5) - (6.12). In the following, we first describe the decision variables. Subsequently, we take a detailed look at the model, its objective function, and constraints.

Decision variables Our model contains two sets of decision variables. Integer variables $x_{i,j} | i, j \in A$ correspond to our repositioning decisions and denote the number of vehicles repositioned from area i to j. Coverage variables $c_{i,j} | i, j \in A$ denote the coverage that is provided by vehicle resources located in or repositioning to i for forecasted trip demand in j. This concept of provided coverage is central to our model. We assume that any vehicle may serve multiple trip requests over the forecast horizon in the neighborhood N_i of its current or assigned location in area i. Recall from the prior section that this neighborhood is defined as the set of areas reachable within the maximum waiting time w_r of a newly arising trip request. This means that a vehicle situated in i could reach such a trip request in the neighborhood on time. Figure 6.2 shows an example of N_i in the city of Hamburg. Given these variable definitions, our model will reposition vehicles to provide coverage in areas with lacking supply. Figure 6.3 shows a simple example in which one vehicle is repositioned from area k to area i in order to cover demand in the two neighboring areas h and j. The precise number of requests that a single vehicle may cover depends on its location and the time of day. Therefore, we adaptively determined parameter $\hat{rs}_i$ as described in the prior Section 6.3.4.

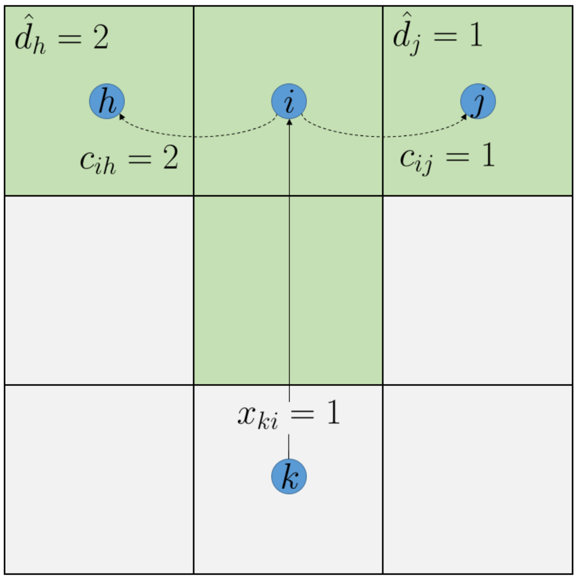


Figure 6.3: Illustration of decision variables. Green areas form the neighborhood of i. Only non-zero demands and relevant area centers are shown for simplicity. In this example one vehicle is repositioned from area k to area i to cover forecasted demand in areas j and h.

Model The objective function (6.5) follows three hierarchical goals which are reflected in the terms of the objective function:

1. Maximize the sum of covered demand, weighted by w^{cov} and w_i.
2. Minimize the number of repositioning movements, weighted by weight w^{re}.
3. Minimize travel times for repositioning movements and demand coverage. The latter may be penalized with a factor $w^{tt,cov} >= 1.0$.

Objective precedence is ensured by weights $w^{cov} > w^{re} > 1$. These weights may be determined based on the maximum overall travel time between two areas tt^{max}. We use $w^{cov} = 10 \cdot tt^{max}$ and $w^{re} = tt^{max}$. With this choice of weights, we ensure that one unit of additional covered demand is prioritized over minimizing movements and travel times. The primary objective is to maximize the acceptance rate of future requests by covering predicted demand. The area-specific weight component enables us to prioritize coverage in certain areas. This can be beneficial in situations where not all demand may be covered by the vehicle fleet. Empirically, it has proven useful to set w_i proportional to the fraction of total demand in i. For instance, one could use $w_i = 1 + \frac{\hat{d}_i}{\hat{D}}$ where $\hat{D}$ is equal to the total forecasted demand $\sum_{i \in A} \hat{d}_i$. The secondary objective stems from the operational concern that we want to move as few vehicles as possible. Particularly, we do not want to move any vehicles at all, if the current fleet configuration can cover all forecasted demand. Otherwise, we run the risk of causing oscillating vehicle movements where vehicles are repeatedly repositioned due to minor shifts in the forecasted demand. Thus, we penalize the movement of vehicles and ensure that repositioning only takes place if it leads to additional covered demand. The tertiary objective ensures that overall travel times are minimized and leads to suitable vehicles being selected for repositioning. Two travel time factors are taken into account. First, we consider travel times incurred by repositioning decisions $x_{i,j}$. Second, we consider anticipated travel times attached to $c_{i,j}$ variables. The assumption is that a vehicle located at $i \in A$ will have to move to $j \in A$ when a request arises. These anticipated travel times are penalized by a factor $w^{tt,cov} \geq 1$ which rewards moving vehicles closer to the predicted demand. This tends to be beneficial as it reduces customer waiting times and improves vehicle utilization.

$$
\begin{aligned}
\text{(FDR-M)} \quad & \sum_{i \in A} \sum_{j \in A} w^{cov} \cdot w_j \cdot c_{i,j} && && (6.5)\\
& - \sum_{i \in A} \sum_{j \in A \setminus \{i\}} w^{re} \cdot x_{i,j} && && \\
& - \sum_{i \in A} \sum_{j \in A} x_{i,j} \cdot tt_{i,j} - \sum_{i \in A} \sum_{j \in A} w^{tt,cov} \cdot c_{i,j} \cdot tt_{i,j} && \rightarrow \max && \\
\text{s.t.} \quad & \sum_{j \in A} x_{i,j} \leq |K_i^{id}| && i \in A && (6.6)\\
& \sum_{j \in A} c_{j,i} \leq \hat{d}_i && i \in A && (6.7)\\
& \sum_{j \in A} c_{i,j} \leq \sum_{j \in A} x_{j,i} \cdot \hat{rs}_i + S_i^{re} + S_i^{act} && i \in A && (6.8)\\
& c_{i,j} = 0 && i \in A, j \notin N_i && (6.9)\\
& x_{i,j} = 0 && i \in A, j \notin A^{re}, i \neq j && (6.10)\\
& x_{i,j} \in \mathbb{N}_0^+ && i, j \in A && (6.11)\\
& c_{i,j} \in \mathbb{R}_0^+ && i, j \in A && (6.12)
\end{aligned}
$$

Constraints (6.6) guarantee that the number of vehicles repositioned from $i \in A$ does not exceed the number of available idle vehicles. Note that only idle vehicles are available for repositioning. Vehicles that are currently performing a repositioning movement cannot be reassigned to a new target. This is due to operational concerns as we do not want to frequently change repositioning targets of individual vehicles and drivers. A repositioning movement may however be interrupted by the vehicle routing algorithm as presented in Chapter 5. In that case, the vehicle is assigned a route with trip requests and starts serving these immediately. Constraints (6.7) ensure that the maximum provided coverage for a given area i is capped by the forecasted demand $\hat{d}_i$. Inversely, Constraints (6.8) limit the provided coverage from area i to the available supply. This supply is equivalent to the number of trip requests that may be served within the forecast horizon h by vehicle resources in i. In order to calculate the available supply, we consider three types of vehicles. Active vehicles located in i contribute to the active supply S_i^{act}, while vehicles currently repositioning to i provide the repositioning supply S_i^{re}. Lastly, supply provided by idle vehicles is considered through the $x_{j,i}$ variables in the right-hand side of the equation. This term includes vehicles staying idle at i as well as vehicles being selected for repositioning to i. The sum of these vehicles is multiplied with $\hat{rs}_i$, which denotes the expected number of trip requests that a vehicle in i will serve within the forecast horizon. This parameter as well as the active and repositioning supply may differ significantly depending on the considered area, current vehicle schedules, and time of day among other factors. Therefore, we introduced an adaptive parameter tuning approach in Section 6.3.4. Constraints (6.9) limit the spatial extent of provided coverage to the given neighborhood N_i as explained earlier. As repositioning is only allowed to a set of target areas A^{re}, Constraints (6.10) ensure that we only send idle vehicles to such areas or leave them at their current location. Lastly, variable domains are given by Constraints (6.11) and (6.12).

6.3.6 Repositioning Target Assignment

As the final step of our repositioning algorithm, we must derive specific repositioning assignments from the aggregated decisions taken in FDR-M. For this purpose, we start by sampling a set of repositioning targets T^{re}. For each area $j \in A$, we compute the sum of repositioning movements towards j as $\sum_{i \in A} x_{i,j}$ and randomly sample that many targets from the potential repositioning locations RL_j and add them to T^{re}. Subsequently, we assign idle vehicles to these targets in a way that minimizes repositioning travel times. We do this by solving the MIP model given in Equations 6.13 – 6.16.

$$\sum_{k \in K^{id}} \sum_{t \in T^{re}} a_{k,t} tt_t^k \qquad \rightarrow \min \tag{6.13}$$

$$\text{s.t.} \quad \sum_{k \in K^{id}} a_{k,t} = 1 \qquad t \in T^{re} \tag{6.14}$$

$$\sum_{t \in T^{re}} a_{k,t} \leq 1 \qquad k \in K^{id} \tag{6.15}$$

$$a_{k,t} \in \{0,1\} \qquad k \in K^{id}, t \in T^{re} \tag{6.16}$$

The decision variables $a_{k,t}$ denote whether a specific vehicle $k \in K^{id}$ is assigned to a repositioning target $t \in T^{re}$. The objective function 6.13 minimizes the sum of travel times for repositioning movements. Constraints 6.14 ensure that exactly one idle vehicle is assigned to each repositioning target. Due to the fact that FDR-M considers the supply of idle vehicles when determining how many vehicles to reposition, it is guaranteed that sufficient idle vehicles are available. Constraints 6.15 on the other hand guarantee that each idle vehicle is assigned to at most one repositioning target. The domain of variables $a_{k,t}$ is given in Constraints 6.16.

6.4 Computational Evaluation

In this section, we evaluate FDR on the same real-world datasets as used in Chapter 5. To perform these evaluations, we use the simulation framework presented in Chapter 4. In the following, we begin by discussing our experimental design in Section 6.4.1 followed by the computational results in Section 6.4.2.

6.4.1 Experimental Design and Setup

In our computational evaluations we investigate the performance of our repositioning approach with regard to the following main goals:

- Show that FDR can be used successfully on large-scale instances in real-time.
- Compare the performance of FDR to a reactive benchmark algorithm REACT and illustrate that non-myopic repositioning leads to performance improvements.
- Assess the robustness of our algorithm and particularly the adaptive parameter tuning process under a variety of demand patterns in order to show that FDR may be easily applied to new application settings.
- Evaluate the necessary forecast quality in order to utilize FDR effectively. For this purpose, we compare a setting with perfect information to one with a naive forecast. The latter simply assumes that demand stays constant and uses the actual demand of the previous horizon as a forecast for the next one.

The datasets and simulation instances used in our evaluations are described in Section 6.4.1.1. The different studied simulation and algorithm settings are detailed in Sections 6.4.1.2 and 6.4.1.3. As in Chapter 5, our planning algorithms and simulation were implemented in C++. We utilize Gurobi 9.1.2 as a MIP solver for FDR-M. All computational studies were run on a computer with an Intel i7-6600U CPU and 20 GB of RAM.

6.4.1.1 Dataset and Simulation Instances

We use the same datasets and instances as in the evaluations of Chapter 5. Hence, for a detailed description, we refer the reader to Sections 5.6.1.1 and 5.6.1.2. For the sake of convenience we repeat the main information concerning the used instances in Table 6.5. Recall that the preliminary instances are used for determining adequate parameters while the main instances are utilized for our actual computational evaluations. We also use the same base fleet sizes as in Chapter 5. These are repeated in Table 6.6. For a discussion concerning the choice of these values, we refer the reader to Section 5.6.1.2.

6.4.1.2 Simulation Settings

In our simulation studies, we evaluate scenarios with different settings as given in Table 6.7. Unless noted otherwise, the default values indicated in bold are used. In the same manner as in the previous chapter, we consider scenarios with short, medium and long time windows controlled by the parameters w_r, L^{min}, m^{det}. These evaluations are presented in Section 6.4.2.5. Additionally, we also evaluate scenarios with different vehicle fleets. In Section 6.4.2.6, we vary the size of the fleet via the vehicle factor that determines the number of vehicles in combination with the base fleet sizes given in the previous section. We also evaluate different vehicle capacities in Section 6.4.2.7.

Table 6.5: Preliminary and main instances.

Group	Dataset	Weekday	Date	# trip requests	Name
Preliminary	CH	Wed	09 Nov 2016	224,219	P-CH-Wed
	HH	Wed	13 Mar 2019	16,158	P-HH-Wed
	MANH	Wed	09 Mar 2016	335,929	P-MANH-Wed
	NYC	Wed	09 Mar 2016	429,855	P-NYC-Wed
Main	CH	Wed	16 Nov 2016	239,037	M-CH-Wed
		Sun	20 Nov 2016	237,037	M-CH-Sun
	HH	Wed	20 Mar 2019	13,556	M-HH-Wed
		Sun	24 Mar 2019	10,669	M-HH-Sun
	MANH	Wed	16 Mar 2016	297,457	M-MANH-Wed
		Sun	20 Mar 2016	269,346	M-MANH-Sun
	NYC	Wed	16 Mar 2016	376,526	M-NYC-Wed
		Sun	20 Mar 2016	368,508	M-NYC-Sun

Table 6.6: Base fleet size per dataset.

	CH	HH	MANH	NYC
Fleet size	1260	90	670	1180

Table 6.7: Simulation settings and potential values. Default values are indicated in **bold**.

Parameter	Notation	Unit	Values
Maximum waiting time	w_r	s	180, **300**, 600
Minimum allowed detour	L^{min}	s	100, **150**, 300
Maximum detour factor	m^{det}	-	1.33, **1.5**, 2
Vehicle factor	-	-	0.8, 0.9, **1.0**, 1.1, 1.2
Vehicle capacity	-	-	2, 3, **4**, 5, 6

6.4.1.3 Algorithm Parameters

In Table 6.8, we summarize the different algorithm parameters studied in this section. Concerning parameters for our vehicle routing algorithm, we use the best-found values from Chapter 5. In this section, we focus purely on parameters pertaining to the repositioning algorithm. For most evaluations in this section, we utilize DIS+LS as our routing algorithm. The interactions between repositioning and routing with and without local search are investigated in detail in Section 6.4.2.4. Throughout this section, the adaptive parameter calculation process from Section 6.3.4 is generally used. To illustrate

Table 6.8: Algorithm parameters and potential values. Default values are indicated in **bold**.

Parameter	Notation	Unit	Values
Routing mode	-	-	DIS, **DIS+LS**
Repositioning mode	-	-	REACT, **FDR**
Adaptive parameters	-	-	no, **yes**
Grid cell side length	g_{re}^{size}	m	1000, 1500, 2000, 3000, 5000
Forecast horizon	h	min	5, 15, 30, 60, 120
Repositioning interval	I	s	30, 60, 120, 180, 300
Obj. weight coverage times	$w^{tt,cov}$	-	0.0, 0.9, 1.0, 1.1, 2.0
Min. vehicles in neighborhood	k_N^{min}	-	1, 3, 5, 10, 20
Proportional area weight	w_i	-	true, false

Table 6.9: Performance indicators.

KPI	Unit	Description
Rej	%	Trip request rejection rate
Wait	s	Avg. customer waiting time
Ride	s	Avg. customer ride time
TT^v	min	Avg. total vehicle travel time
TT_{req}^v	s	Avg. vehicle travel time per served trip request
RT	min	Total running time
RT^{dis}	min	Total running time for the dispatching algorithm
RT_r^{dis}	min	Avg. running time for dispatching one trip request
RT^{ls}	min	Total running time for the local search algorithm
RT^{re}	min	Total running time for the repositioning algorithm
RT^o	min	Total running time for other tasks

its impact, we also perform evaluations with fixed parameters and compare the results in Section 6.4.2.9. Concerning the six algorithm parameters $g_{re}^{size}, h, I, w^{tt,cov}, k_N^{min}$ and w_i, we perform experiments on our preliminary instances and determine adequate values in Section 6.4.2.1.

6.4.1.4 Forecast Mode

An important factor for our repositioning approach FDR is the demand forecast that is used as an input. Generally, in this section, we assume a *perfect* demand forecast. The reason for this is that we focus on the evaluation of our repositioning algorithm itself and illustrate its potential performance with a good forecast. However, to show that our algorithm already performs well with relatively simple forecasting models, we perform evaluations with a *naive* forecast. The utilized forecast and the results are discussed in Section 6.4.2.8. Together, these two forecast modes form an upper and lower bound on the forecast quality that a state-of-the-art forecasting model would yield.

6.4.1.5 Performance Indicators

We use the same performance indicators as in Chapter 5. For the sake of convenience, these are repeated in Table 6.9. For more detailed explanations of the individual KPIs, we refer the reader to Section 5.6.1.5.

6.4.2 Computational Results

In the following sections, we discuss our main computational results and findings. We start by presenting the results of our parameter studies on preliminary instances in Section 6.4.2.1. Subsequently, we take a look at running times and aggregated results with default settings in Sections 6.4.2.2 and 6.4.2.3. In the remaining Sections 6.4.2.4 through 6.4.2.9, we present a detailed look into the impact of factors such as time windows, fleet sizes, or the adaptive parameter calculation process.

6.4.2.1 Parameter Influence

In this section, we study the impact of several key algorithm parameters to determine adequate values for the remainder of the experiments. The considered parameters and values are given in Table 6.10. We do not perform a full grid search of all parameter combinations as this would be too computationally expensive. Instead, we vary each parameter individually and leave the other ones at their default values. We focus on the impact of each parameter on the total running time and the trip request rejection rate.

Average results on our perliminary test instances are illustrated in Figure 6.4. The main takeaways are the following:

- Reducing the repositioning interval I leads to improved rejection rates which is to be expected as it enables us to regularly adjust repositioning movements to reflect the current system state. Due to the minor influence on the running time, we can opt for the smallest value of 30 seconds.

Table 6.10: Algorithm parameters with default starting values denoted in *italic* and the best found values in **bold**.

Parameter	Notation	Unit	Values
Grid cell side length	g_{re}^{size}	m	1000, 1500, *2000*, 3000, **5000**
Forecast horizon	h	m	5, **15**, *30*, 60, 120
Repositioning interval	I	s	**30**, 60, *120*, 180, 300
Obj. weight coverage times	$w^{tt,cov}$	-	0.0, 0.9, *1.0*, 1.1, **2.0**
Min. vehicles in neighborhood	k_N^{min}	-	1, 3, *5*, 10, **20**
Proportional area weight	w_i	-	*true*, false

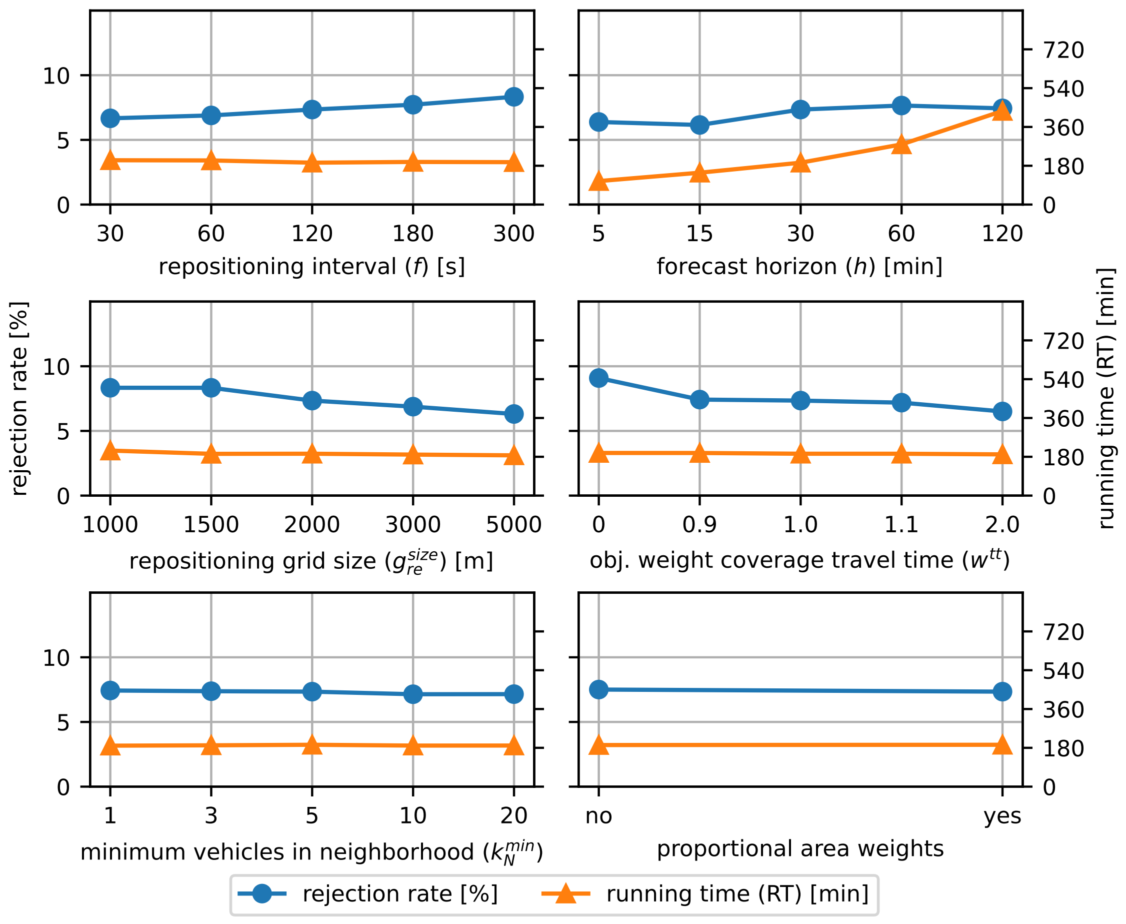


Figure 6.4: Influence of the repositioning interval (I), grid cell cell side length (g_{re}^{size}), minimum vehicles in neighborhood (k_N^{min}), forecast horizon (k^{max}), objective weight for coverage travel times ($w^{tt,cov}$), and proportional area objective weights.

- Concerning the forecast horizon h, a value of 15 minutes leads to the best rejection rates. A shorter forecast horizon leaves us too little time to react to vehicle shortages. On the other hand, a longer forecast horizon leads to the consideration of forecasted requests which may not be immediately relevant for deciding repositioning movements. In addition, an increase in the forecast horizon leads to a steady increase in running time. This is mostly due to the necessary data collection for the adaptive parameter tuning process and could be improved by more efficient data structures.
- The grid cell side length g_{re}^{size} of 5000 m x 5000 m leads to the best rejection rates, illustrating that a certain level of aggregation in the demand forecasts and decision-making is beneficial to the overall performance and there is no benefit in using a finer spatial resolution. The impact on the running time is negligible although a larger grid cell size also leads to slightly reduced running times.
- Regarding the objective weights for coverage travel times $w^{tt,cov}$ it is beneficial to penalize these in the objective function compared to the travel times for repositioning

movements. This encourages the model to reposition vehicles closer to the forecasted demand.

- The minimum number of vehicles in the neighborhood k_N^{min} only has a very minor impact on both the rejection rate and the running time. The best results are achieved with a value of 20. Hence, this setting is used for the remaining experiments.
- Regarding the objective weights for covering specific areas w_i, we evaluated two different settings. One in which areas are all weighted equally and one in which coverage is prioritized in high-demand areas by setting weights proportional to the demand. The latter leads to slightly better results although the differences are minor.

6.4.2.2 Running Times

Table 6.11 shows the average running times for our two repositioning modes REACT and FDR. The main conclusion is that with FDR activated, we are still able to process large-scale simulation scenarios faster than real-time. In fact, the repositioning algorithm itself, which is executed every 30 seconds, only makes up a negligible portion of the overall running time. We still see a large increase in running time when activating FDR compared to the runs with REACT. This is mainly due to one major factor. As we may observe, the running time RT^o for other tasks besides the core planning functionalities exhibits the largest difference. This is caused by the handling of additional data needed as input for the repositioning algorithm and particularly the adaptive parameter calculation. To facilitate this adaptive mechanism, we store detailed histories of vehicle itineraries. The running time could be improved by using more efficient data structures for this purpose and preventing unnecessary copy operations. However, even with the current implementation,

Table 6.11: Overview of average running times. Rows denoted as "ALL" contain averages across all four datasets.

Data	**Mode**	**RT^{re} [min]**	**RT^{dis} [min]**	**RT^{ls} [min]**	**RT^{o} [min]**	**RT [min]**
CH	REACT	1.20	27.30	35.67	20.25	84.42
	FDR	0.58	26.58	36.21	69.61	132.98
HH	REACT	0.00	0.22	0.02	0.47	0.71
	FDR	0.19	0.30	0.02	1.40	1.91
MANH	REACT	0.12	38.98	31.23	18.31	88.64
	FDR	0.39	43.28	31.07	71.01	145.75
NYC	REACT	1.79	60.06	69.38	36.07	167.30
	FDR	0.95	67.38	68.57	171.42	308.32
ALL	REACT	0.78	31.64	34.08	18.78	85.27
	FDR	0.53	34.39	33.97	78.36	147.24

our approach is applicable to large instances on the most challenging dataset (NYC). The total running time still constitutes a major speed-up compared to the simulated timespan of 1800 minutes.

6.4.2.3 Results on Default Scenarios

Table 6.12 summarizes our main computational results with our repositioning algorithm FDR and the benchmark approach reactive repositioning algorithm (REACT). The main takeaway is that forecast-driven repositioning manages to achieve an average reduction in trip request rejection rate of 4.81 percentage points. Additionally, it leads to reduced waiting times (-19.48 s) and slightly reduced ride times (-7.07 s) which would improve customer satisfaction. These improvements come at the price of an increased TT^v_{req}. This is to be expected as additional repositioning movements lead to less efficient routes. Particularly in the case of the HH dataset, FDR enables us to serve remote trip requests that would otherwise have to be rejected. However, as these remote requests are inefficient to serve, the overall efficiency of our routes is decreased. We believe that this trade-off is still worthwhile as on average TT^v_{req} is merely increased by 19.85 s.

Figure 6.5 presents a more detailed look at the occupation of the vehicle fleet with REACT and FDR for instance M-NYC-Wed. As we can see, FDR manages to prevent the rejection of trip requests throughout most of the day. During the evening demand peak, there are rejections due to the overloaded vehicle fleet. In the morning there is a minor amount of rejections which could potentially be attributed to an underestimation of the necessary vehicle resources.

Table 6.12: Aggregated results with REACT and FDR. Rows denoted as “ALL” contain averages across all four datasets.

Data	**Mode**	**Rej** [%]	**Wait** [s]	**Ride** [s]	$\mathbf{TT}^v$ [min]	$\mathbf{TT}^v_{req}$ [s]	**RT** [min]
CH	REACT	10.77	200.48	593.16	929.52	**349.81**	84.42
	FDR	**5.37**	**182.94**	**579.67**	1019.92	361.97	132.98
HH	REACT	17.55	174.95	476.19	842.71	**456.72**	0.71
	FDR	**7.37**	**155.67**	**472.67**	1061.37	512.51	1.91
MANH	REACT	6.39	222.96	293.07	1028.68	**156.16**	88.64
	FDR	**5.09**	**203.75**	**290.38**	1056.15	158.16	145.75
NYC	REACT	9.19	216.69	375.70	1037.41	**217.17**	167.30
	FDR	**6.86**	**194.81**	**367.10**	1110.35	226.63	308.32
ALL	REACT	10.98	203.77	434.53	959.58	**294.97**	85.27
	FDR	**6.17**	**184.29**	**427.46**	1061.95	314.82	147.24

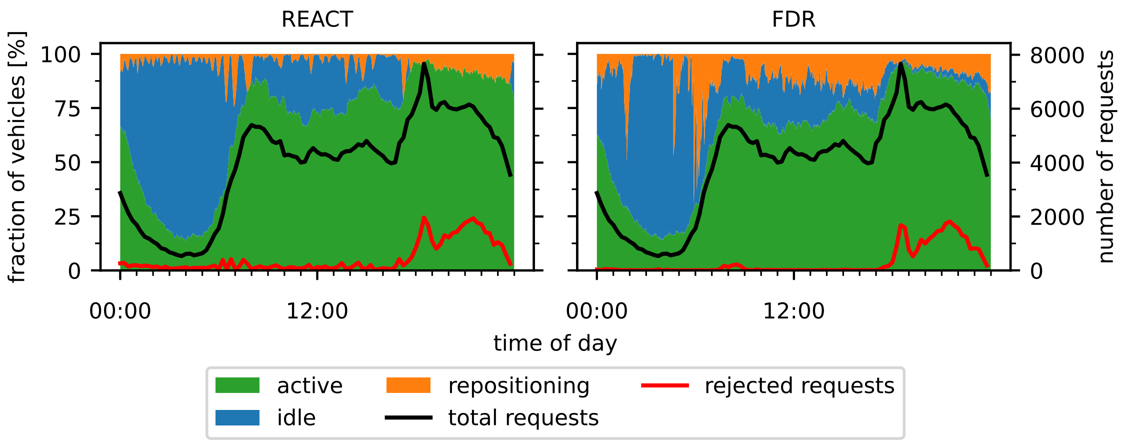


Figure 6.5: Vehicle states with REACT and FDR for instance M-NYC-Wed.

6.4.2.4 Impact of the Local Search

While most experiments in this section were ran with our routing algorithm DIS+LS, we also ran simulations without the local search in order to gain some insights into the interaction of both algorithms. Figure 6.6 shows the most interesting result from these evaluations. We ran our instances with all four combinations of our repositioning algorithms REACT and FDR as well as the vehicle routing algorithms DIS and DIS+LS. Figure 6.6 illustrates the average rejection rates obtained from these simulation runs. As we can see, the impact of the local search is larger when using our forecast-driven repositioning approach. On average, activating the local search yields an improvement in the rejection rate of 0.6 percentage points with REACT versus 1.4 percentage points with FDR. These results illustrate that the two approaches benefit from being used in combination. We assume that the reason for this is that FDR improves the positioning of our vehicle fleet and the local search can better utilize these improved positions.

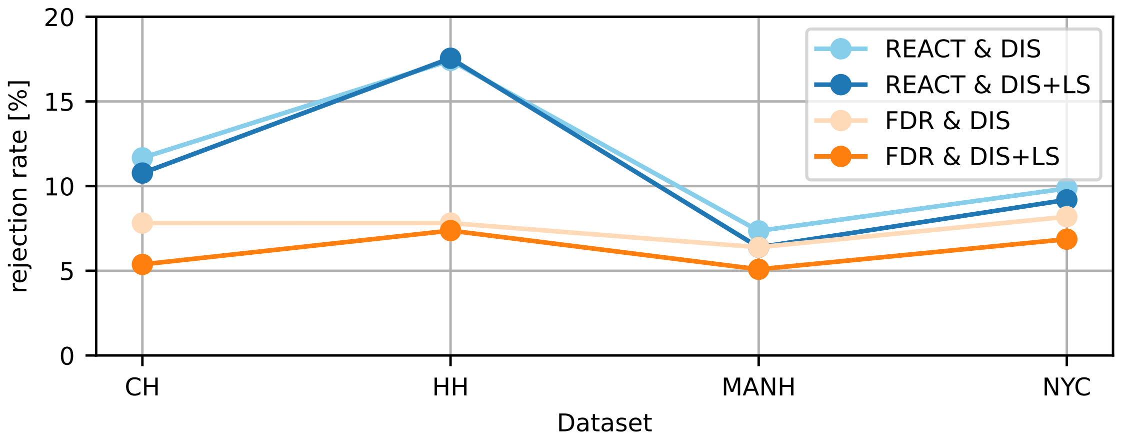


Figure 6.6: Rejection rates for different datasets and combinations of vehicle routing and repositioning.

Table 6.13: Time window settings.

	Unit	Short	Medium	Long
w_r	s	180	300	600
L^{min}	s	100	150	300
m^{det}	-	1.33	1.5	2.0

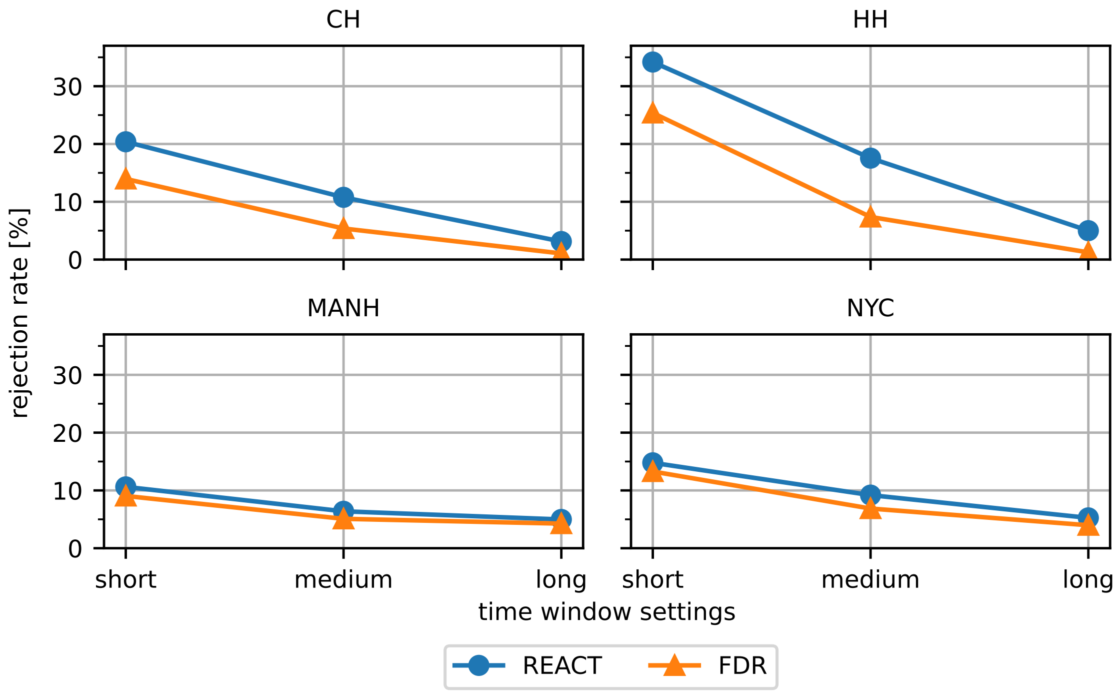


Figure 6.7: Rejection rates for different datasets and time windows.

6.4.2.5 Time Windows

As in Chapter 5, we evaluate different time window settings for the trip requests. Table 6.13 summarizes our three settings with short, medium and long time windows. The average rejection rates for the different scenarios and datasets are depicted in Figure 6.7. The results illustrate that regardless of the time window length, FDR achieves improved results compared to REACT. The absolute improvement is lowest with the long time window setting at an average of 1.95 percentage points in contrast to 4.58 and 4.8 percentage points with short and medium time windows respectively. This is to be expected as with long time windows the rejection rates are already relatively low with REACT.

6.4.2.6 Fleet Size

In our evaluations, we assess the impact of the vehicle fleet size by running scenarios with a vehicle factor of $0.8 - 1.2$. This factor is multiplied by the base fleet size as given in

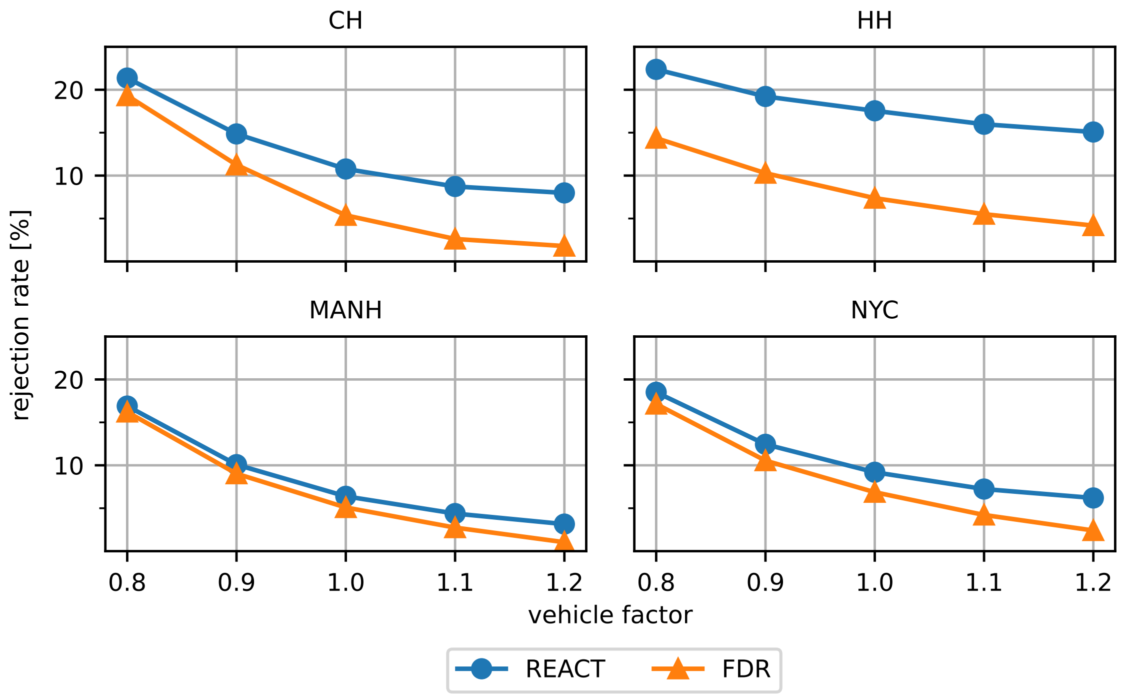


Figure 6.8: Rejection rates for different datasets and vehicle factors.

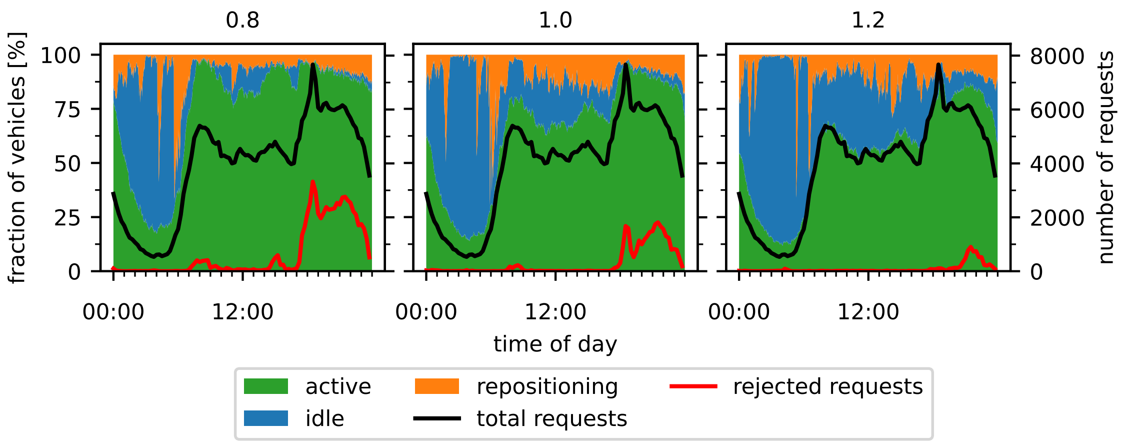


Figure 6.9: Vehicle states with different vehicle factors for instance M-NYC-Wed.

Section 6.4.1.1. Our main findings are illustrated by the rejection rates for different vehicle factors in Figure 6.8. The central takeaway is that the gap in rejection rate between REACT and FDR grows with a larger fleet size. FDR is able to exploit the additional vehicles to serve additional trip requests. In contrast, with a small vehicle fleet, a large portion of the fleet is occupied at all times leaving little room for improvement through repositioning. At many points in time, there may not even be available vehicles for repositioning as all vehicles are occupied. This is further illustrated by Figure 6.9 that visualizes the vehicle states for scenario M-NYC-Wed and different fleet factors. It shows that with a vehicle factor of 0.8, the complete fleet is occupied during the morning and evening demand peaks and throughout most of the day. Hence, not enough vehicles are available for repositioning. In contrast, with the 1.2 vehicle factor, there are idle vehicles available at most times. Only during the evening peak, there is a period of time at which all vehicles are either active or repositioning.

6.4.2.7 Vehicle Capacity

Figure 6.10 shows trip request rejection rates for different vehicle capacities ranging from 2 – 6 passengers. As expected, the overall rejection rate decreases as the vehicle capacity is increased. However, there are diminishing returns and, for instance, when increasing the capacity from 5 to 6, we only see minor improvements. This can mainly be attributed to the customer time windows as it becomes increasingly difficult to combine additional trip requests in one route while still adhering to their time window constraints. When comparing the two repositioning modes REACT and FDR, we observe that the gap between modes slightly widens as the capacity is increased. This is a similar effect as with the fleet size in the previous section. With FDR we are able to use the increased capacity more effectively as vehicles are better positioned to serve incoming trip requests and we have a reduced lead time when approaching new requests.

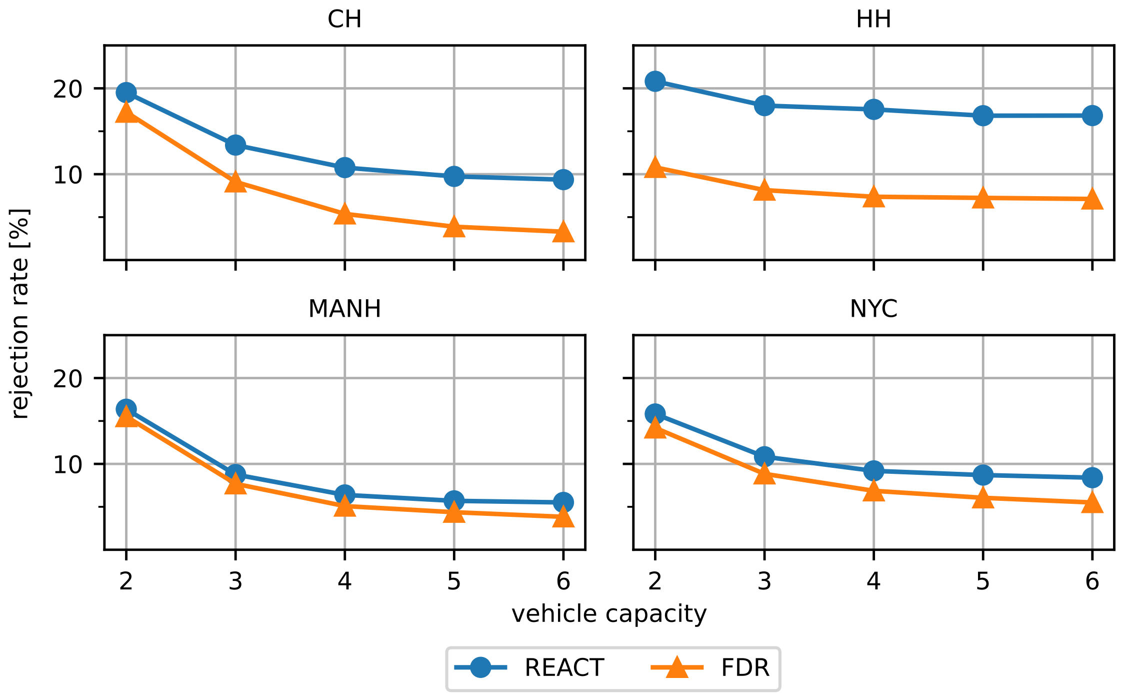


Figure 6.10: Rejection rates for different datasets and vehicle capacities.

6.4.2.8 Utilizing a Naive Demand Forecast

Throughout most of this section, we have combined FDR with a perfect demand forecast to illustrate the potential of our approach with a good forecast quality. In practice, one only has access to a flawed forecast generated by a forecasting model. In this section we combine FDR with a naive forecast, to show that our approach already performs well with a very simple forecasting model.

Naive demand forecast Our naive demand forecast assumes that demand stays constant, i.e. the forecasted demand $\hat{d}_i$ for an area i over the next h^+ minutes is equal to the actual demand within the previous h^- minutes. When working with a short forecast horizon as in this work, this simple forecast already provides a reasonable forecast quality. To assess the quality of the forecast, we replicated the evaluations by Yao et al. (2019) on the NYC taxi dataset. They use the time period between 10 February 2015 and 1 March 2015 as their test data and split the region into 200 grid cells of 1x1 km. Their forecast horizon is 30 minutes. We aimed to reproduce these settings as accurately as possible, however, there may be slight differences in the experimental setup due to data preprocessing, cleaning, and the exact spatial coverage of the grid. Nevertheless, we believe that our results should be roughly comparable to those obtained in their original paper and therefore give the reader a good idea of how the naive forecast performs compared to the forecasting models evaluated in Yao et al. (2019). Table 6.14 summarizes these results and reports the root-mean-square error (RMSE) for the naive forecast compared to the methods from Yao et al. (2019). As the results show, the naive forecast falls between traditional time series methods such as historical averages or ARIMA and classical machine learning approaches like gradient boosting. The best results are obtained by deep learning approaches utilizing CNNs and LSTMs. For details concerning the evaluated methods, we refer the reader to the original paper. The main takeaway for this work is that the naive forecast delivers an adequate quality for short-term demand predictions.

Computational results with the naive forecast Table 6.15 summarizes the results with the perfect and naive forecast modes. The main conclusion is that FDR may already be successfully used with a simple demand forecasting model such as our naive forecast. On average, FDR performs only slightly worse with the naive forecast concerning the trip request rejection rate and all other performance indicators merely exhibit minor deviations. On one dataset, we unexpectedly even achieve better results with the naive forecast. We mainly attribute this effect to the structure of the forecast as the naive forecast underestimates future demand when it is rising (e.g. in the morning) and overestimates it when it is declining. Sometimes, this may lead to better overall results as FDR may miscalculate how many vehicles are needed to serve the arising demand. A more in-depth evaluation of these cases could lead to some potential improvements in our repositioning approach.

Table 6.14: Comparison of the naive forecast with established forecasting methods. *Naive* indicates the RMSE of the model used in this work, while the reference models are from the paper by Yao et al. (2019).

Model	**RMSE**
HA	43.82
ARIMA	36.53
LR	28.51
MLP	26.67
XGBoost	26.07
LinUOTD	28.48
ConvLSTM	28.13
DeepSD	26.35
ST-ResNet	26.23
DMVST-Net	25.74
STDN	24.10
Naive	35.54

Table 6.15: Aggregated results with the perfect and naive forecast. Rows denoted as "ALL" contain averages across all four datasets.

Data	**Forecast Mode**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT$^v_{req}$ [s]**	**RT [min]**
CH	perfect	5.37	182.94	**579.67**	1019.92	361.97	132.98
	naive	**5.28**	**182.06**	579.76	1013.27	**359.23**	132.05
HH	perfect	**7.37**	**155.67**	472.67	1061.37	**512.51**	1.91
	naive	7.88	155.73	**472.28**	1071.40	521.05	1.88
MANH	perfect	**5.09**	203.75	290.38	1056.15	**158.16**	145.75
	naive	5.14	**202.90**	**290.23**	1055.79	158.21	146.32
NYC	perfect	**6.86**	194.81	367.10	1110.35	**226.63**	308.32
	naive	6.72	**192.93**	**366.85**	1112.91	226.82	304.38
ALL	perfect	**6.17**	184.29	427.46	1061.95	**314.82**	147.24
	naive	6.26	**183.41**	**427.28**	1063.34	316.33	146.16

6.4.2.9 Impact of Adaptive Parameter Calculation

As a final evaluation in this section, we take a look at the impact of the adaptive parameter tuning approach described in Section 6.3.4. It adaptively calculates the number of trip requests $\hat{rs}_i$ that a vehicle located in area i is expected to serve within the next forecast horizon. This approach serves two main purposes. Firstly, it removes the necessity for tuning parameters a priori. Hence, one could start the ride-sharing service in a new area of operations without the need for prior data to determine algorithm parameters. Secondly,

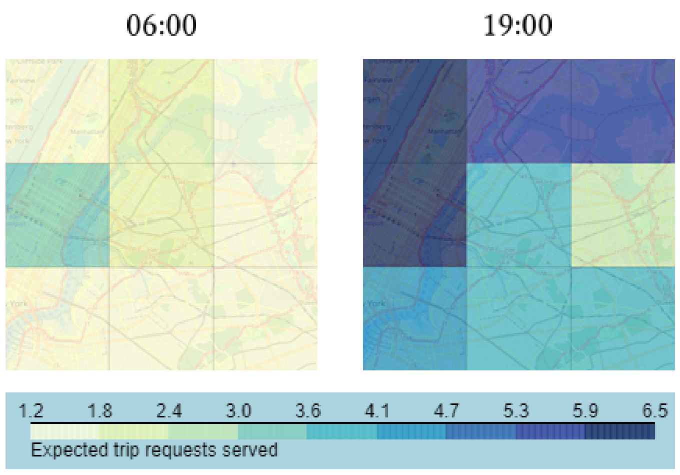


Figure 6.11: Expected requests served $\hat{rs}_i$ within the next 15 minutes for an excerpt of NYC on 16 March 2016 at 06:00 and 19:00.

Table 6.16: Fixed values of $\hat{rs}_i$ for different datasets.

	CH	HH	MANH	NYC
$\hat{rs}_i$	2.30	2.06	5.17	3.54

the adaptive approach should more accurately reflect temporal and spatial differences in the number of trip requests a vehicle may be expected to serve. This should in turn improve the overall performance of our ride-sharing system. The usefulness of estimating the number of trip requests served depending on the time of day and geographical location is illustrated in Figure 6.11. It shows values for $\hat{rs}_i$ for areas in NYC at 06:00 and 19:00. Clearly, there are large differences in $\hat{rs}_i$. On the one hand, values at 06:00 are overall lower compared to 19:00 as the total demand at this point in time is lower. This makes it difficult to efficiently combine multiple trip requests into one route and leads to fewer expected requests served per vehicle. On the other hand, we can also observe geographical differences. The values for $\hat{rs}_i$ tend to be higher for the Manhattan areas (towards the left) as the demand density is higher.

To assess whether the adaptive parameter tuning process can adequately consider these effects, we ran experiments in which $\hat{rs}_i$ is fixed to a single value. This value was determined based number of trip requests that vehicles served in our preliminary studies. The values on the different datasets are given in Table 6.16. The results with fixed parameters and adaptive parameter tuning are compared in Table 6.17. Regarding the rejection rate, the adaptive mechanism clearly outperforms the fixed parameter setting. This is to be expected as the fixed setting will often miscalculate the number of vehicles that are needed at a given time in a specific location. In theory, one could determine time- and location-dependent fixed parameters. However, we believe that an adaptive mechanism as proposed in this work represents a more applicable and flexible option. It manages to consider shifts in demand without the need for prior data-intensive tuning and can react to structural changes in demand on the fly.

Table 6.17: Aggregated results with fixed parameters and adaptive calculation. Rows denoted as "ALL" contain averages across all four datasets.

Data	**Parameter Mode**	**Rej [%]**	**Wait [s]**	**Ride [s]**	**TT^v [min]**	**TT$^v_{req}$ [s]**	**RT [min]**
CH	fixed	5.82	186.11	580.20	981.96	**350.12**	133.03
	adaptive	**5.37**	**182.94**	**579.67**	1019.92	361.97	132.98
HH	fixed	9.33	158.25	**471.10**	998.31	**492.33**	1.89
	adaptive	**7.37**	**155.67**	472.67	1061.37	512.51	1.91
MANH	fixed	5.24	209.76	291.07	1041.11	**156.15**	146.44
	adaptive	**5.09**	**203.75**	**290.38**	1056.15	158.16	145.75
NYC	fixed	8.60	203.17	**365.25**	1021.60	**212.49**	309.73
	adaptive	**6.86**	**194.81**	367.10	1110.35	226.63	308.32
ALL	fixed	7.25	189.32	**426.91**	1010.75	**302.77**	147.77
	adaptive	**6.17**	**184.29**	427.46	1061.95	314.82	147.24

6.5 Conclusions

In this chapter, we have presented a new approach for idle vehicle repositioning in the context of dynamic ride-sharing services. The central component of our algorithm is a MIP model that aims to balance supply provided by the vehicle fleet and expected demand given by a forecast. We include an adaptive parameter tuning process in order to reflect temporal and spatial changes in the number of trip requests a vehicle is expected to serve. Our approach is evaluated with simulation scenarios on four real-world datasets. Our results show that the algorithm facilitates a robust improvement in trip request rejection rate over a diverse set of scenarios. Moreover, we are able to show that it already performs well with a simple naive demand forecast and can therefore be easily applied in practical settings with little available data.

In the future, we see the potential to apply our forecast-driven repositioning approach to other application settings such as the repositioning of autonomous guided vehicles (AGVs) in a large-scale industrial manufacturing or warehousing setting. In addition, one particularly interesting research direction would be the transfer of our approach to applications with decentrally controlled vehicles such as MOD services like Uber or Lyft with self-employed drivers. In these settings, our model and algorithm could be adapted to modify prices and incentivize drivers to reposition to certain areas. Moreover, it appears promising to integrate additional forecasting components into our system in order to adequately consider other sources of uncertainty. One big factor here would be the inclusion of a forecasting model for travel times, as these vary substantially depending on the current time of day, location, and traffic situation. There are also several minor extensions and improvements that could be made for our MIP model. Despite our consideration of the current system state and the adaptive parameter tuning process, additional details could still be added in several areas. For instance, it could prove beneficial to consider the current vehicle routes which contain information regarding the future vehicle location and occupation. Moreover, our parameter tuning currently only considers information about the immediate past and could be replaced by a more complex approach. For example, one could use a trained machine learning model that forecasts the expected number of served trip requests per vehicle. Besides modifying the current algorithm, we also think it is promising to solve a similar repositioning approach with completely different algorithms. While the running times and performance of our algorithm are good, the need for a commercial-grade MIP solver can be problematic in real-world use cases. Therefore, one could apply different approaches such as network flow models which may be able to solve a similar repositioning model considering current supply and forecasted demand.

7 Conclusion and Outlook

This thesis presented detailed insights into the planning of dynamic ride-sharing services. In Chapters 2 and 3, we introduced ride-sharing in the broader context of mobility-as-a-service and gave an overview of planning problems that arise when operating a ride-sharing service. Subsequently, Chapters 4 - 6 presented a simulation-based evaluation framework and operational planning algorithms for vehicle routing and repositioning. In this chapter, we summarize our main contributions and results and propose some promising directions for future research.

7.1 Summary and Conclusion

Modern mobility-on-demand applications such as ride-sharing, ride-hailing, and bike-sharing have revolutionized the urban mobility landscape throughout recent years. This thesis focused on dynamic ride-sharing services as a model that can potentially offer flexible mobility-on-demand to customers while at the same time efficiently utilizing vehicle resources and minimizing the impact on urban traffic congestion and pollution. In our work, we considered the particular challenges that arise in the dynamic ride-sharing setting, such as the high degree of dynamism and the large amount of trip requests to process. With these challenges in mind, we aimed to develop a comprehensive toolset to plan and evaluate the operation of a dynamic ride-sharing service.

In Chapter 2, we presented a taxonomy of mobility-as-a-service offerings, comparing ride-sharing to other MaaS applications such as car-sharing, ride-hailing and car-pooling. We emphasized the differences between ride-sharing and these other mobility offerings, for instance, the vehicle-sharing aspect when comparing car-sharing and ride-sharing. This taxonomy of MaaS applications was followed up by a more detailed discussion of the application setting of dynamic ride-sharing in Chapter 3. We highlighted key characteristics of ride-sharing services such as the high degree of dynamism, the large number of trip requests, and the importance of different business models in the field. Moreover, we analyze the different stakeholders of a ride-sharing system such as the service operator, customers, drivers, and also the city or region in which the service operates. Based on these stakeholders and their different objectives, we derived a set of strategical, tactical, and operational planning problems that are relevant for ride-sharing providers. On the strategical level, we considered infrastructure and fleet design decisions. However, we believe that these planning problems are similar to ones arising in other application settings, for instance, facility location problems. Therefore, we think that existing solution approaches for these problems may be utilized. On the tactical level, we discussed the fleet and personnel scheduling problems. Similar to the strategical level, we believe that comparable problems have been studied sufficiently in other application domains such as public transport planning and the solution approaches may be adapted for dynamic ride-sharing. On the operational planning level, we see some unique challenges due to the dynamic nature of the ride-sharing service and the potentially large number of simultaneous trip requests in the system. Hence, we consider the operational planning of ride-sharing services as a focus in this thesis. Besides presenting the planning tasks associated with ride-sharing services, Chapter 3 also considered the execution of shared rides and how it interacts with the planning level, for instance, in case of disruptions. This is particularly important when simulating the execution of shared rides to evaluate the operational planning approaches.

This introduction into the application context of ride-sharing was followed up in Chapter 4 by our proposed system architecture for the operational planning and simulation-based evaluation of ride-sharing systems. We reviewed related literature and found two main groups of simulation approaches for dynamic ride-sharing. On the one hand, there are approaches based on micro- or mesoscopic traffic simulations. These enable detailed simulation studies of urban mobility and can also include other mobility modes such as personal vehicles or public transport. However, they are comparatively difficult to set up and configure. On the other hand, there are simulation tools that focus purely on the simulation of the ride-sharing service and do not include detailed modeling of overall mobility demand and traffic behavior. We proposed our own simulation framework SimDRS that falls into the second category. It simulates vehicles and customer requests in a ride-sharing service and is capable of efficiently running large-scale simulation scenarios. Additionally, it only needs a minimal amount of data concerning trip requests, the vehicle fleet and the road network to get started. We envision that our framework may be utilized for multiple purposes. Due to its modular nature, it enables algorithm developers to evaluate their operational planning approaches for dynamic ride-sharing. We utilize this in Chapters 5 and 6 to evaluate our own algorithms for vehicle routing and repositioning. Moreover, the lightweight simulation setup allows ride-sharing providers to quickly evaluate simulation scenarios in cities or regions to which they want to expand their service. In addition, the simulation framework is easily extended and can accommodate additional data sources such as time-dependent travel times if they are available.

Chapter 5 focuses on the vehicle routing problem for dynamic ride-sharing and introduces a heuristic solution approach that consists of a fast dispatching algorithm for new trip requests and a local search improvement phase that leverages available computational time to improve the current routing plan. The approach is evaluated on diverse scenarios based on real world-data from Chengdu, Hamburg and New York City. Our computational results show that the application of our local search improvement phase reduces the number of rejected trip requests by up to 40 % and the vehicle travel times per request by up to 8 % depending on the specific simulation scenario. This illustrates the potential benefits of utilizing an improvement phase in conjunction with a fast dispatching heuristic to facilitate an improved solution quality with nearly instantaneous responses to customer requests. Moreover, our results show that the usage of ride-sharing can indeed reduce the generated vehicle traffic compared to the usage of private vehicles. This shows the potential for ride-sharing to deliver flexible mobility while reducing traffic congestion.

Chapter 6 considers the problem of idle vehicle repositioning. The main objective is to reposition idle vehicles in a way that anticipates future trip request demand. For this purpose, we present a forecast-driven repositioning algorithm (FDR). This algorithm uses a demand forecast as an input for a mathematical model that determines repositioning decisions. The model considers vehicle supply and forecasted trip request demand and tries to balance these two aspects in a way that maximizes the acceptance of trip requests while minimizing the operational cost of repositioning movements. To consider spatial and temporal differences in the number of trip requests that a vehicle can be expected to serve, we introduce an adaptive parameter calculation approach that automatically adapts key algorithm parameters to the current system state. The forecast-driven repositioning model is integrated into a real-time planning approach and evaluated on simulation scenarios. As a benchmark for our algorithm, we use a reactive repositioning scheme. Our algorithm was evaluated on the same real-world instances from Chengdu, Hamburg and New York City as in Chapter 5. The computational results show that FDR manages to improve trip request rejection rates by an average of 44 % while also reducing waiting and ride times of customers. These results illustrate that intelligent repositioning can lead to a significantly better performance of the ride-sharing system. Moreover, our algorithm performs well under diverse circumstances and, due to its adaptive nature, may be easily applied to new scenarios.

Overall, this thesis has presented the planning context of dynamic ride-sharing systems with relevant planning problems, stakeholders, and characteristics. On the operational planning level, we have designed algorithms for vehicle routing and repositioning. To evaluate these algorithms, we have proposed a simulation-based evaluation framework that allowed us to assess the performance of our approaches under a diverse set of simulation scenarios based on real-world data. Our evaluations have shown that our algorithms can be used for large-scale ride-sharing applications and exhibit robust performance in diverse scenarios. In particular, they do not rely on large amounts of prior data and can be easily applied to new application settings. We believe that the structured analysis of the application context of dynamic ride-sharing presented in this thesis combined with the operational planning and evaluation framework forms a solid foundation for planning and operating ride-sharing services. This serves as the basis for future work concerning the operation of ride-sharing systems.

7.2 Outlook

This thesis has mainly focused on the operational planning of dynamic ride-sharing services and evaluated the proposed algorithms through simulations. In this regard, we see two promising directions for extensions. Firstly, it would be interesting to also consider tactical planning problems such as vehicle and staff scheduling as these are coupled with the operational planning level and form the basis for the operational decisions. While we do not think that the application setting of ride-sharing introduces any completely new problem aspects on the tactical level compared to existing staff and vehicle scheduling problems, we still consider it beneficial to move towards a more holistic planning approach akin to other domains such as public transport (Desaulniers and Hickman 2007). Secondly, we consider it highly relevant to evaluate the interaction between the execution of rides and operational planning in more detail. While some fundamental considerations concerning this interaction were presented in this thesis, we see a multitude of promising research directions. For instance, the simulation framework could be extended in a way that makes it possible to model several types of disruptions such as vehicle breakdowns and customer cancellations. Based on these disruptions, one could integrate special disruption handling techniques into the planning service and extend the vehicle routing algorithm to also trigger the re-routing of vehicles. Besides disruption handling, it would also be relevant to leverage the data gathered from the execution more effectively. In this work, we use the data for demand forecasting and adaptive parameter calculation. There are several other potential uses for the collected data such as the estimation of vehicle travel times or the usage on a tactical planning level to determine an adequate vehicle fleet size throughout the day.

Besides the scope of the planning and simulation framework, there are naturally also several ways to extend the presented operational planning algorithms. The vehicle routing solution is relatively simple in nature and could be improved in several ways. In our opinion, the cheapest insertion dispatching heuristic is well-suited for its job to find insertion positions for new trip requests very quickly. However, the local search improvement phase could be extended in multiple ways. For instance, there is a large amount of existing work on search operators and search algorithms for vehicle routing problems that could be adapted to the dynamic planning setting of dynamic ride-sharing. This could improve the solution quality compared to the results from this work as we only worked with simple search operators. Besides improving the algorithm itself, we also see several application-specific requirements that could be considered in more advanced ways in our approach. We discussed the integration with the execution level concerning disruption handling and re-routing above. Besides this, we consider the integration of pre-booked trip requests or some form of combined static and dynamic planning to be a promising direction for future work. In many application settings, it is highly relevant to consider a portion of the trip requests in advance and make commitments toward the customer. To improve the robustness and reliability of routing plans, we envision the usage of more data-driven approaches. Particularly promising is the estimation of realistic travel times as this would lead to fewer delays and disruptions during the execution of routes.

Concerning the task of vehicle repositioning, our algorithm might be improved in several ways. For instance, the adaptive parameter tuning process only considers information about past trip requests and routes. It would also be possible to include data from forecasts and future vehicle routes. In general, the currently planned vehicle routes could be incorporated into our mathematical model in some form to utilize this information instead of only using the vehicle locations at the time the model is solved. One drawback of our algorithm is that it relies on a commercial grade mixed-integer programming solver. We think that a similar modeling approach as ours could be used in a network flow formulation, removing the need for a solver. Another direction of research would be the usage of our repositioning model to other application domains. In this focus, we focused solely on the use case of dynamic ride-sharing. However, repositioning problems arise in many other settings such as ambulance repositioning or the repositioning of autonomous guided vehicles in manufacturing and warehousing. We believe that the general principles behind our model could be transferred to other domains.

Lastly, we believe that a detailed study of decentralized approaches and a comparison to centralized algorithms such as the ones presented in this work would be highly relevant. Many large ride-sharing providers such as Uber and Lyft rely on decentralized and incentive-based approaches to control their drivers due to their business model. Our simulation framework could be extended to include decentral driver decisions. For instance, drivers should have the option to reject assigned routes and repositioning would be based on price incentives. On the planning side, the central repositioning approach would be replaced by pricing algorithms. In theory, centralized algorithms should perform better than decentralized ones as they have complete control over drivers and vehicles. In our view, a structured quantitative comparison of decentral and central control mechanisms for repositioning would be a promising future research topic.

Bibliography

Adnan, M., F. Pereira, C. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva (2016). SimMobility: a multiscale integrated agent-based simulation platform. *Transportation Research Board 95th Annual Meeting*.

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* 223 (2), pp. 295–303. DOI: 10.1016/j.ejor.2012.05.028.

Aimsun (2022). Aimsun Ride. Accessed 12 Aug 2022. https://www.aimsun.com/de/aimsun-ride-forschungsprogramm/.

Alonso-Mora, J., S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus (2017a). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* 114 (3), pp. 462–467. DOI: 10.1073/pnas.1611675114.

Alonso-Mora, J., A. Wallar, and D. Rus (2017b). Predictive routing for autonomous mobility-on-demand systems with ride-sharing. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Vancouver, BC: IEEE, pp. 3583–3590. DOI: 10.1109/IROS.2017.8206203.

American Public Transportation Association (2007). *Public Transportation Fact Book, 2007.* Tech. rep.

Attanasio, A., J.-F. Cordeau, G. Ghiani, and G. Laporte (2004). Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing* 30 (3), pp. 377–387. DOI: 10.1016/j.parco.2003.12.001.

Auld, J., M. Hope, H. Ley, V. Sokolov, B. Xu, and K. Zhang (2016). POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. *Transportation Research Part C: Emerging Technologies* 64, pp. 101–116. DOI: 10.1016/j.trc.2015.07.017.

Azevedo, C. L. et al. (2016). Microsimulation of Demand and Supply of Autonomous Mobility On Demand. *Transportation Research Record: Journal of the Transportation Research Board* 2564 (1), pp. 21–30. DOI: 10.3141/2564-03.

Baykasoğlu, A., K. Subulan, A. S. Taşan, and N. Dudaklı (2019). A review of fleet planning problems in single and multimodal transportation systems. *Transportmetrica A: Transport Science* 15 (2), pp. 631–697. DOI: 10.1080/23249935.2018.1523249.

Beaudry, A., G. Laporte, T. Melo, and S. Nickel (2010). Dynamic transportation of patients in hospitals. *OR Spectrum* 32 (1), pp. 77–107. DOI: 10.1007/s00291-008-0135-6.

Beaujon, G. J. and M. A. Turnquist (1991). A Model for Fleet Sizing and Vehicle Allocation. *Transportation Science* 25 (1), pp. 19–45. DOI: 10.1287/trsc.25.1.19.

Behrisch, M., L. Bieker, J. Erdmann, and D. Krajzewicz (2011). SUMO – Simulation of Urban MObility: An Overview. *Proceedings of SIMUL 2011*. Ed. by A. Omerovic, D. Simoni, and G. Bobashev. Barcelona: ThinkMind.

Bent, R. and P. Van Hentenryck (2004). Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. *Operations Research* 52 (6), pp. 977–987. DOI: 10.1287/opre.1040.0124.

Bent, R. and P. Van Hentenryck (2007). Waiting and Relocation Strategies in Online Stochastic Vehicle Routing. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Ed. by M. M. Veloso. Hyderabad: Morgan Kaufmann Publishers Inc., pp. 1816–1821.

Berbeglia, G., J.-F. Cordeau, and G. Laporte (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research* 202 (1), pp. 8–15. DOI: 10.1016/j.ejor.2009.04.024.

Bertoli, F., P. Kilby, and T. Urli (2020). A column-generation-based approach to fleet design problems mixing owned and hired vehicles. *International Transactions in Operational Research* 27 (2), pp. 899–923. DOI: 10.1111/itor.12647.

Bertsimas, D. and J. Tsitsiklis (1993). Simulated Annealing. *Statistical Science* 8 (1). DOI: 10.1214/ss/1177011077.

Bischoff, J., M. Maciejewski, and K. Nagel (2017). City-wide shared taxis: A simulation study in Berlin. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Yokohama: IEEE, pp. 275–280. DOI: 10.1109/ITSC.2017.8317926.

Borndörfer, R., M. Grötschel, F. Klostermeier, and C. Küttner (1999). Telebus Berlin: Vehicle Scheduling in a Dial-a-Ride System. *Computer-Aided Transit Scheduling*. Ed. by N. H. M. Wilson. Vol. 471. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 391–422. DOI: 10.1007/978-3-642-85970-0_19.

Boyacı, B., K. G. Zografos, and N. Geroliminis (2017). An integrated optimization-simulation framework for vehicle and personnel relocations of electric carsharing systems with reservations. *Transportation Research Part B: Methodological* 95, pp. 214–237. DOI: 10.1016/j.trb.2016.10.007.

Bruglieri, M., F. Pezzella, and O. Pisacane (2019). An Adaptive Large Neighborhood Search for relocating vehicles in electric carsharing services. *Discrete Applied Mathematics* 253. DOI: 10.1016/j.dam.2018.03.067.

Burke, E. K., P. De Causmaecker, G. V. Berghe, and H. Van Landeghem (2004). The State of the Art of Nurse Rostering. *Journal of Scheduling* 7 (6), pp. 441–499. DOI: 10.1023/B:JOSH.0000046076.75950.0b.

Casas, J., J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday (2010). Traffic Simulation with Aimsun. *Fundamentals of Traffic Simulation.* Ed. by J. Barceló. Vol. 145. New York, NY: Springer New York, pp. 173–232. DOI: 10.1007/978-1-4419-6142-6_5.

Castiglione, J. and D. Cooper (2018). TNCs and Congestion. Accessed 11 Mar 2020. https://www.sfcta.org/projects/tncs-and-congestion.

Cats, O., R. Kucharski, S. R. Danda, and M. Yap (2022). Beyond the dichotomy: How ride-hailing competes with and complements public transport. *PLOS ONE* 17 (1). DOI: 10.1371/journal.pone.0262496.

Chan, N. D. and S. A. Shaheen (2012). Ridesharing in North America: Past, Present, and Future. *Transport Reviews* 32 (1), pp. 93–112. DOI: 10.1080/01441647.2011.621557.

Chen, L., Y. Gao, Z. Liu, X. Xiao, C. S. Jensen, and Y. Zhu (2018). PTrider: a price-and-time-aware ridesharing system. *Proceedings of the VLDB Endowment* 11 (12), pp. 1938–1941. DOI: 10.14778/3229863.3236229.

Cheng, P., H. Xin, and L. Chen (2017). Utility-Aware Ridesharing on Road Networks. *Proceedings of the 2017 ACM International Conference on Management of Data.* Chicago, IL: Association for Computing Machinery, pp. 1197–1210. DOI: 10.1145/3035918.3064008.

Chow, J. and J. Jung (2019). Large-Scale Simulation-Based Evaluation of Fleet Repositioning Strategies for Dynamic Rideshare in New York City. *SAE Technical Papers.* DOI: 10.4271/2019-01-0924.

Cordeau, J.-F. (2006). A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Operations Research* 54 (3), pp. 573–586. DOI: 10.1287/opre.1060.0283.

Cordeau, J.-F. and G. Laporte (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37 (6), pp. 579–594. DOI: 10.1016/S0191-2615(02)00045-0.

Cordeau, J.-F. and G. Laporte (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research* 153 (1), pp. 29–46. DOI: 10.1007/s10479-007-0170-8.

Cordeau, J.-F., G. Laporte, and S. Ropke (2008). Recent Models and Algorithms for One-to-One Pickup and Delivery Problems. *The Vehicle Routing Problem: Latest Advances and New Challenges.* Ed. by B. Golden, S. Raghavan, and E. Wasil. Vol. 43. Boston, MA: Springer US, pp. 327–357. DOI: 10.1007/978-0-387-77778-8_15.

Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research* 122 (2), pp. 272–288. DOI: 10.1016/S0377-2217(99)00233-7.

Daskin, M. S., ed. (2013). *Network and Discrete Location: Models, Algorithms, and Applications, Second Edition.* Hoboken, NJ: John Wiley & Sons, Inc. DOI: 10.1002/9781118537015.

Deb, S., K. Tammi, K. Kalita, and P. Mahanta (2018). Review of recent trends in charging infrastructure planning for electric vehicles. *Wiley Interdisciplinary Reviews: Energy and Environment* 7 (6). DOI: 10.1002/wene.306.

Desaulniers, G. and M. D. Hickman (2007). Chapter 2 Public Transit. *Handbooks in Operations Research and Management Science.* Ed. by C. Barnhart and G. Laporte. Vol. 14. Elsevier, pp. 69–127. DOI: 10.1016/S0927-0507(06)14002-5.

Dibbelt, J., B. Strasser, and D. Wagner (2016). Customizable Contraction Hierarchies. *Journal of Experimental Algorithmics* 21 (1), pp. 1–49. DOI: 10.1145/2886843.

Didi Chuxing (2020). KDD Cup 2020 – Learning to Dispatch and Reposition on a Mobility-on-Demand Platform. Accessed 16 Jul 2020. https://outreach.didichuxing.com/appEn-vue/KDD_CUP_2020.

Doubek, J. (2018). New York City Temporarily Halts More Uber And Lyft Cars On The Road. Accessed 11 Mar 2020. https://www.npr.org/2018/08/09/637008474/new-york-city-temporarily-halts-more-uber-and-lyft-cars-on-the-road.

Drexl, M. and M. Schneider (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research* 241 (2), pp. 283–308. DOI: 10.1016/j.ejor.2014.08.030.

Eglese, R. and S. Zambirinis (2018). Disruption management in vehicle routing and scheduling for road freight transport: a review. *TOP* 26 (1), pp. 1–17. DOI: 10.1007/s11750-018-0469-4.

Engelhardt, R., F. Dandl, and K. Bogenberger (2020). Speed-up Heuristic for an On-Demand Ride-Pooling Algorithm. DOI: 10.48550/arXiv.2007.14877.

Engelhardt, R., F. Dandl, A.-A. Syed, Y. Zhang, F. Fehn, F. Wolf, and K. Bogenberger (2022). FleetPy: A Modular Open-Source Simulation Tool for Mobility On-Demand Services. DOI: 10.48550/ARXIV.2207.14246.

Ernst, A., H. Jiang, M. Krishnamoorthy, and D. Sier (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research* 153 (1), pp. 3–27. DOI: 10.1016/S0377-2217(03)00095-X.

Etezadi, T. and J. E. Beasley (1983). Vehicle Fleet Composition. *Journal of the Operational Research Society* 34 (1), pp. 87–91. DOI: 10.1057/jors.1983.11.

Fagnant, D. J. and K. M. Kockelman (2018). Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas. *Transportation* 45 (1), pp. 143–158. DOI: 10.1007/s11116-016-9729-z.

Ferrero, F., G. Perboli, M. Rosano, and A. Vesco (2018). Car-sharing services: An annotated review. *Sustainable Cities and Society* 37, pp. 501–518. DOI: 10.1016/j.scs.2017.09.020.

Ferrucci, F., S. Bock, and M. Gendreau (2013). A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. *European Journal of Operational Research* 225 (1), pp. 130–141. DOI: 10.1016/j.ejor.2012.09.016.

Fishman, E., S. Washington, and N. Haworth (2013). Bike Share: A Synthesis of the Literature. *Transport Reviews* 33 (2), pp. 148–165. DOI: 10.1080/01441647.2013.775612.

Fitzsimmons, E. G. (2017). Subway Ridership Declines in New York. Is Uber to Blame? Accessed 25 Oct 2022. https://www.nytimes.com/2017/02/23/nyregion/new-york-city-subway-ridership.html.

Funke, B., T. Grünert, and S. Irnich (2005). Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration. *Journal of Heuristics* 11 (4), pp. 267–306. DOI: 10.1007/s10732-005-1997-2.

Gambella, C., E. Malaguti, F. Masini, and D. Vigo (2018). Optimizing relocation operations in electric car-sharing. *Omega* 81, pp. 234–245. DOI: 10.1016/j.omega.2017.11.007.

Gehrke, S. R., A. Felix, and T. G. Reardon (2019). Substitution of Ride-Hailing Services for More Sustainable Travel Options in the Greater Boston Region. *Transportation Research Record: Journal of the Transportation Research Board* 2673 (1), pp. 438–446. DOI: 10.1177/0361198118821903.

Glover, F. and M. Laguna (1998). Tabu Search. *Handbook of Combinatorial Optimization.* Ed. by D.-Z. Du and P. M. Pardalos. Boston, MA: Springer US, pp. 2093–2229. DOI: 10.1007/978-1-4613-0303-9_33.

Golalikhani, M., B. B. Oliveira, M. A. Carravilla, J. F. Oliveira, and A. P. Antunes (2021). Carsharing: A review of academic literature and business practices toward an integrated decision-support framework. *Transportation Research Part E: Logistics and Transportation Review* 149, p. 102280. DOI: 10.1016/j.tre.2021.102280.

Groër, C., B. Golden, and E. Wasil (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation* 2 (2), pp. 79–101. DOI: 10.1007/s12532-010-0013-5.

Gurumurthy, K. M., F. de Souza, A. Enam, and J. Auld (2020). Integrating Supply and Demand Perspectives for a Large-Scale Simulation of Shared Autonomous Vehicles. *Transportation Research Record: Journal of the Transportation Research Board* 2674 (7), pp. 181–192. DOI: 10.1177/0361198120921157.

Hansen, P. and N. Mladenović (2003). Variable Neighborhood Search. *Handbook of Metaheuristics*. Ed. by F. Glover and G. A. Kochenberger. Vol. 57. Boston, MA: Springer US, pp. 145–184. DOI: 10.1007/0-306-48056-5_6.

Hoff, A., H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen (2010). Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research* 37 (12), pp. 2041–2061. DOI: 10.1016/j.cor.2010.03.015.

Horni, A., K. Nagel, and K. Axhausen, eds. (2016). The Multi-Agent Transport Simulation MATSim. London: Ubiquity Press. DOI: 10.5334/baw.

Huang, H., M. Pouls, A. Meyer, and M. Pauly (2020a). Travel Time Prediction Using Tree-Based Ensembles. *Computational Logistics*. Ed. by E. Lalla-Ruiz, M. Mes, and S. Voß. Vol. 12433. Cham: Springer International Publishing, pp. 412–427. DOI: 10.1007/978-3-030-59747-4_27.

Huang, K., K. An, J. Rich, and W. Ma (2020b). Vehicle relocation in one-way station-based electric carsharing systems: A comparative study of operator-based and user-based methods. *Transportation Research Part E: Logistics and Transportation Review* 142. DOI: 10.1016/j.tre.2020.102081.

Huang, Y., F. Bastani, R. Jin, and X. S. Wang (2014). Large scale real-time ridesharing with service guarantee on road networks. *Proceedings of the VLDB Endowment* 7 (14), pp. 2017–2028. DOI: 10.14778/2733085.2733106.

Ibarra-Rojas, O. J., F. Delgado, R. Giesen, and J. C. Muñoz (2015). Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological* 77, pp. 38–75. DOI: 10.1016/j.trb.2015.03.002.

Ichoua, S., M. Gendreau, and J.-Y. Potvin (2006). Exploiting Knowledge About Future Demands for Real-Time Vehicle Dispatching. *Transportation Science* 40 (2), pp. 211–225. DOI: 10.1287/trsc.1050.0114.

Jittrapirom, P., V. Caiati, A.-M. Feneri, S. Ebrahimigharehbaghi, M. J. A. González, and J. Narayan (2017). Mobility as a Service: A Critical Review of Definitions, Assessments of Schemes, and Key Challenges. *Urban Planning* 2 (2), pp. 13–25. DOI: 10.17645/up.v2i2.931.

Jorge, D. and G. Correia (2013). Carsharing systems demand estimation and defined operations: a literature review. *European Journal of Transport and Infrastructure Research* 13 (3). DOI: 10.18757/EJTIR.2013.13.3.2999.

Jung, J., R. Jayakrishnan, and J. Y. Park (2016). Dynamic Shared-Taxi Dispatch Algorithm with Hybrid-Simulated Annealing: Dynamic shared-taxi dispatch algorithm. *Computer-Aided Civil and Infrastructure Engineering* 31 (4), pp. 275–291. DOI: 10.1111/mice.12157.

Kagerbauer, M., N. Kostorz, G. Wilkes, F. Dandl, R. Engelhardt, U. Glöckl, E. Fraedrich, and F. Zwick (2021). Ridepooling in der Modellierung des Gesamtverkehrs - Methodenbericht zur MOIA Begleitforschung. DOI: 10.5445/IR/1000141282.

Ke, J., H. Zheng, H. Yang, Xiqun, and Chen (2017). Short-Term Forecasting of Passenger Demand under On-Demand Ride Services: A Spatio-Temporal Deep Learning Approach. *Transportation Research Part C: Emerging Technologies* 85, pp. 591–608. DOI: 10.1016/j.trc.2017.10.016. arXiv: 1706.06279.

Koç, Ç., T. Bektaş, O. Jabali, and G. Laporte (2016). The impact of depot location, fleet composition and routing on emissions in city logistics. *Transportation Research Part B: Methodological* 84, pp. 81–102. DOI: 10.1016/j.trb.2015.12.010.

Kucharski, R. and O. Cats (2022). Simulating two-sided mobility platforms with MaaSSim. *PLOS ONE* 17 (6). DOI: 10.1371/journal.pone.0269682.

Lai, D. S. W. and J. M. Y. Leung (2018). Real-time rescheduling and disruption management for public transit. *Transportmetrica B: Transport Dynamics* 6 (1), pp. 17–33. DOI: 10.1080/21680566.2017.1358678.

Li, B., D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang (2011). Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops).* Seattle, WA: IEEE, pp. 63–68. DOI: 10.1109/PERCOMW.2011.5766967.

Li, X., G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang (2012). Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science* 6 (1), pp. 111–121. DOI: 10.1007/s11704-011-1192-6.

Liao, S., L. Zhou, X. Di, B. Yuan, and J. Xiong (2018). Large-scale short-term urban taxi demand forecasting using deep learning. *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jeju: IEEE, pp. 428–433. DOI: 10.1109/ASPDAC.2018.8297361.

Lippi, M., M. Bertini, and P. Frasconi (2013). Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning. *IEEE Transactions on Intelligent Transportation Systems* 14 (2), pp. 871–882. DOI: 10.1109/TITS.2013.2247040.

Lokhandwala, M. and H. Cai (2020). Siting charging stations for electric vehicle adoption in shared autonomous fleets. *Transportation Research Part D: Transport and Environment* 80. DOI: 10.1016/j.trd.2020.102231.

Lowalekar, M., P. Varakantham, and P. Jaillet (2019). ZAC: A Zone Path Construction Approach for Effective Real-Time Ridesharing. *Proceedings of the International Conference on Automated Planning and Scheduling* 29 (1), pp. 528–538. DOI: 10.1609/icaps.v29i1.3519.

Lowalekar, M., P. Varakantham, and P. Jaillet (2021). Zone pAth Construction (ZAC) based Approaches for Effective Real-Time Ridesharing. *Journal of Artificial Intelligence Research* 70, pp. 119–167. DOI: 10.1613/jair.1.11998.

Lyft (2022). Scheduled rides for riders. Accessed 21 Oct 2022. https://help.lyft.com/hc/e/all/articles/115013078668-Scheduled-rides-for-riders.

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

Ma, S., Y. Zheng, and O. Wolfson (2015). Real-Time City-Scale Taxi Ridesharing. *IEEE Transactions on Knowledge and Data Engineering* 27 (7), pp. 1782–1795. DOI: 10.1109/TKDE.2014.2334313.

Mallig, N., M. Kagerbauer, and P. Vortisch (2013). mobiTopp – A Modular Agent-based Travel Demand Modelling Framework. *Procedia Computer Science* 19, pp. 854–859. DOI: 10.1016/j.procs.2013.06.114.

Melo, M. T., S. Nickel, and F. Saldanha-da-Gama (2009). Facility location and supply chain management – A review. *European Journal of Operational Research* 196 (2), pp. 401–412. DOI: 10.1016/j.ejor.2008.05.007.

Mes, M., M. van der Heijden, and P. Schuur (2010). Look-ahead strategies for dynamic pickup and delivery problems. *OR Spectrum* 32 (2), pp. 395–421. DOI: 10.1007/s00291-008-0146-3.

Mitrović-Minić, S. and G. Laporte (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological* 38 (7), pp. 635–655. DOI: 10.1016/j.trb.2003.09.002.

MOIA (2022a). MOIA Servicegebiet. Accessed 01 Aug 2022. https://help.moia.io/hc/de/articles/360000988738-Servicegebiet-Hamburg-.

MOIA (2022b). What are MOIA stops? Accessed 21 Oct 2022. https://help.moia.io/hc/en-us/articles/360000792145-What-are-MOIA-stops-.

Molenbruch, Y., K. Braekers, and A. Caris (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research* 259 (1), pp. 295–325. DOI: 10.1007/s10479-017-2525-0.

Nelson, E. and N. Sadowsky (2019). Estimating the Impact of Ride-Hailing App Company Entry on Public Transportation Use in Major US Urban Areas. *The B.E. Journal of Economic Analysis & Policy* 19 (1). DOI: 10.1515/bejeap-2018-0151.

Nourinejad, M. and M. J. Roorda (2014). A dynamic carsharing decision support system. *Transportation Research Part E: Logistics and Transportation Review* 66, pp. 36–50. DOI: 10.1016/j.tre.2014.03.003.

NYC Taxi and Limousine Commission (2022). TLC Trip Record Data. Accessed 17 Oct 2022. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

Ong, H. Y., D. Freund, and D. Crapis (2021). Driver Positioning and Incentive Budgeting with an Escrow Mechanism for Ride-Sharing Platforms. *INFORMS Journal on Applied Analytics* 51 (5), pp. 373–390. DOI: 10.1287/inte.2021.1091.

Parragh, S. N., K. F. Doerner, and R. F. Hartl (2008). A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft* 58 (2), pp. 81–117. DOI: 10.1007/s11301-008-0036-4.

Pillac, V., M. Gendreau, C. Guéret, and A. L. Medaglia (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225 (1), pp. 1–11. DOI: 10.1016/j.ejor.2012.08.015.

Pouls, M., N. Ahuja, K. Glock, and A. Meyer (2022). Adaptive forecast-driven repositioning for dynamic ride-sharing. *Annals of Operations Research*. DOI: 10.1007/s10479-022-04560-3.

Pouls, M., A. Meyer, and N. Ahuja (2020). Idle Vehicle Repositioning for Dynamic Ride-Sharing. *Computational Logistics.* Ed. by E. Lalla-Ruiz, M. Mes, and S. Voß. Vol. 12433. Cham: Springer International Publishing, pp. 507–521. DOI: 10.1007/978-3-030-59747-4_33.

Pouls, M., A. Meyer, and K. Glock (2021). Real-Time Dispatching with Local Search Improvement for Dynamic Ride-Sharing. *Computational Logistics.* Ed. by M. Mes, E. Lalla-Ruiz, and S. Voß. Vol. 13004. Cham: Springer International Publishing, pp. 299–315. DOI: 10.1007/978-3-030-87672-2_20.

Powell, J. W., Y. Huang, F. Bastani, and M. Ji (2011). Towards Reducing Taxicab Cruising Time Using Spatio-Temporal Profitability Maps. *Advances in Spatial and Temporal Databases.* Ed. by D. Pfoser, Y. Tao, K. Mouratidis, M. A. Nascimento, M. Mokbel, S. Shekhar, and Y. Huang. Vol. 6849. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 242–260. DOI: 10.1007/978-3-642-22922-0_15.

Psaraftis, H. N., M. Wen, and C. A. Kontovas (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks* 67 (1), pp. 3–31. DOI: 10.1002/net.21628.

PTV Group (2022a). PTV MaaS Modeller. Accessed 12 Aug 2022. https://www.ptvgroup.com/de/mobilitynext/.

PTV Group (2022b). Verkehrsplanungssoftware PTV Visum. Accessed 12 Aug 2022. https://www.myptv.com/de/mobilitaetssoftware/ptv-visum.

Qurashi, M., H. Jiang, and C. Antoniou (2020). Modeling autonomous dynamic vanpooling services in SUMO by integrating a dynamic routing scheduler. DOI: 10.5281/ZENODO.4955079.

Reiss, S. and K. Bogenberger (2017). A Relocation Strategy for Munich's Bike Sharing System: Combining an operator-based and a user-based Scheme. *Transportation Research Procedia* 22, pp. 105–114. DOI: 10.1016/j.trpro.2017.03.016.

Repoux, M., B. Boyacı, and N. Geroliminis (2015). Simulation and optimization of one-way car-sharing systems with variant relocation policies. *TRB 94th Annual Meeting Compendium of Papers.*

Repoux, M., M. Kaspi, B. Boyacı, and N. Geroliminis (2019). Dynamic prediction-based relocation policies in one-way station-based carsharing systems with complete journey reservations. *Transportation Research Part B: Methodological* 130, pp. 82–104. DOI: 10.1016/j.trb.2019.10.004.

Riley, C., P. van Hentenryck, and E. Yuan (2020). Real-Time Dispatching of Large-Scale Ride-Sharing Systems: Integrating Optimization, Machine Learning, and Model Predictive Control. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial*

Intelligence. Yokohama: International Joint Conferences on Artificial Intelligence Organization, pp. 4417–4423. DOI: 10.24963/ijcai.2020/609.

Riley, C., A. Legrain, and P. Van Hentenryck (2019). Column Generation for Real-Time Ride-Sharing Operations. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Ed. by L.-M. Rousseau and K. Stergiou. Vol. 11494. Cham: Springer International Publishing, pp. 472–487. DOI: 10.1007/978-3-030-19212-9_31.

Ritzinger, U., J. Puchinger, and R. F. Hartl (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54 (1), pp. 215–231. DOI: 10.1080/00207543.2015.1043403.

Ruch, C., S. Horl, and E. Frazzoli (2018). AMoDeus, a Simulation-Based Testbed for Autonomous Mobility-on-Demand Systems. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, pp. 3639–3644. DOI: 10.1109/ITSC.2018.8569961.

Salanova, J. M., M. Estrada, G. Aifadopoulou, and E. Mitsakis (2011). A review of the modeling of taxi services. *Procedia - Social and Behavioral Sciences* 20, pp. 150–161. DOI: 10.1016/j.sbspro.2011.08.020.

Santi, P., G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti (2014). Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences of the United States of America* 111 (37), pp. 13290–13294. DOI: 10.1073/pnas.1403657111.

Schmid, V. (2012). Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research* 219 (3), pp. 611–621. DOI: 10.1016/j.ejor.2011.10.043.

Schmid, V. and J. F. Ehmke (2015). Integrated timetabling and vehicle scheduling with balanced departure times. *OR Spectrum* 37 (4), pp. 903–928. DOI: 10.1007/s00291-015-0398-7.

Schneider, M. and M. Drexl (2017). A survey of the standard location-routing problem. *Annals of Operations Research* 259 (1), pp. 389–414. DOI: 10.1007/s10479-017-2509-0.

Shah, S., M. Lowalekar, and P. Varakantham (2020). Neural Approximate Dynamic Programming for On-Demand Ride-Pooling. *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (1), pp. 507–515. DOI: 10.1609/aaai.v34i01.5388.

Shaheen, S. and A. Cohen (2019). Shared ride services in North America: definitions, impacts, and the future of pooling. *Transport Reviews* 39 (4), pp. 427–442. DOI: 10.1080/01441647.2018.1497728.

Shaheen, S. A., S. Guzman, and H. Zhang (2010). Bikesharing in Europe, the Americas, and Asia: Past, Present, and Future. *Transportation Research Record: Journal of the Transportation Research Board* 2143 (1), pp. 159–167. DOI: `10.3141/2143-20`.

Shareef, H., M. M. Islam, and A. Mohamed (2016). A review of the stage-of-the-art charging technologies, placement methodologies, and impacts of electric vehicles. *Renewable and Sustainable Energy Reviews* 64, pp. 403–420. DOI: `10.1016/j.rser.2016.06.033`.

Shen, S., Z.-Q. Wei, L.-J. Sun, Y.-Q. Su, R.-C. Wang, and H.-M. Jiang (2018). The Shared Bicycle and Its Network—Internet of Shared Bicycle (IoSB): A Review and Survey. *Sensors* 18 (8). DOI: `10.3390/s18082581`.

Su, Y., K. Xie, H. Wang, Z. Liang, W. Art Chaovalitwongse, and P. M. Pardalos (2021). Airline Disruption Management: A Review of Models and Solution Methods. *Engineering* 7 (4), pp. 435–447. DOI: `10.1016/j.eng.2020.08.021`.

SUMO (2022a). SUMO – Activity-based Demand Generation. Accessed 12 Aug 2022. `https://sumo.dlr.de/docs/Demand/Activity-based_Demand_Generation.html`.

SUMO (2022b). SUMO – Taxi. Accessed 10 Aug 2022. `https://sumo.dlr.de/docs/Simulation/Taxi.html`.

Syed, A. A., F. Dandl, B. Kaltenhäuser, and K. Bogenberger (2021). Density Based Distribution Model for Repositioning Strategies of Ride Hailing Services. *Frontiers in Future Transportation* 2, p. 681451. DOI: `10.3389/ffutr.2021.681451`.

Tafreshian, A., N. Masoud, and Y. Yin (2020). Frontiers in Service Science: Ride Matching for Peer-to-Peer Ride Sharing: A Review and Future Directions. *Service Science* 12 (2), pp. 44–60. DOI: `10.1287/serv.2020.0258`.

Thomas, B. W. (2007). Waiting Strategies for Anticipating Service Requests from Known Customer Locations. *Transportation Science* 41 (3), pp. 319–331.

Tirachini, A. (2020). Ride-hailing, travel behaviour and sustainable mobility: an international review. *Transportation* 47 (4), pp. 2011–2047. DOI: `10.1007/s11116-019-10070-2`.

TomTom (2022). Traffic Data & Traffic Stats. Accessed 12 Aug 2022. `https://www.tomtom.com/products/traffic-stats/`.

Tong, Y., Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu (2018). A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment* 11 (11), pp. 1633–1646. DOI: `10.14778/3236187.3236211`.

Uber (2020). Surge pricing. Accessed 07 Apr 2020. `https://marketplace.uber.com/pricing/surge-pricing`.

Uber (2021). UberPool. Accessed 08 Jan 2021. https://www.uber.com/us/en/ride/uberpool.

Uber (2022). Get your ride right with Uber Reserve. Accessed 21 Oct 2022. https://www.uber.com/us/en/ride/how-it-works/reserve/.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and M. Hennig (2019). Offline–Online Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests. *Transportation Science* 53 (1), pp. 185–202. DOI: 10.1287/trsc.2017.0767.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and B. W. Thomas (2020). On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics* 9 (2). DOI: 10.1016/j.ejtl.2020.100008.

Ulmer, M. W., N. Soeffker, and D. C. Mattfeld (2018). Value function approximation for dynamic multi-period vehicle routing. *European Journal of Operational Research* 269 (3), pp. 883–899. DOI: 10.1016/j.ejor.2018.02.038.

Utriainen, R. and M. Pöllänen (2018). Review on mobility as a service in scientific publications. *Research in Transportation Business & Management* 27, pp. 15–23. DOI: 10.1016/j.rtbm.2018.10.005.

Van den Bergh, J., J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck (2013). Personnel scheduling: A literature review. *European Journal of Operational Research* 226 (3), pp. 367–385. DOI: 10.1016/j.ejor.2012.11.029.

Vlahogianni, E. I., J. C. Golias, and M. G. Karlaftis (2004). Short-term traffic forecasting: Overview of objectives and methods. *Transport Reviews* 24 (5), pp. 533–557. DOI: 10.1080/0144164042000195072.

Vlahogianni, E. I., M. G. Karlaftis, and J. C. Golias (2014). Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies* 43, pp. 3–19. DOI: 10.1016/j.trc.2014.01.005.

Voccia, S. A., A. M. Campbell, and B. W. Thomas (2019). The Same-Day Delivery Problem for Online Purchases. *Transportation Science* 53 (1), pp. 167–184. DOI: 10.1287/trsc.2016.0732.

Wageningen-Kessels, F. van, H. van Lint, K. Vuik, and S. Hoogendoorn (2015). Genealogy of traffic flow models. *EURO Journal on Transportation and Logistics* 4 (4), pp. 445–473. DOI: 10.1007/s13676-014-0045-5.

Wang, B., H. Liang, S. Hörl, and F. Ciari (2017). Dynamic ride sharing implementation and analysis in MATSim. DOI: 10.3929/ETHZ-B-000183727.

Wickert, T. I., P. Smet, and G. Vanden Berghe (2019). The nurse rerostering problem: Strategies for reconstructing disrupted schedules. *Computers & Operations Research* 104, pp. 319–337. DOI: 10.1016/j.cor.2018.12.014.

Wilkes, G., R. Engelhardt, L. Briem, F. Dandl, P. Vortisch, K. Bogenberger, and M. Kagerbauer (2021). Self-Regulating Demand and Supply Equilibrium in Joint Simulation of Travel Demand and a Ride-Pooling Service. *Transportation Research Record: Journal of the Transportation Research Board* 2675 (8), pp. 226–239. DOI: 10.1177/0361198121997140.

Xu, M. and Q. Meng (2019). Fleet sizing for one-way electric carsharing services considering dynamic vehicle relocation and nonlinear charging profile. *Transportation Research Part B: Methodological* 128, pp. 23–49. DOI: 10.1016/j.trb.2019.07.016.

Yang, Q. and H. N. Koutsopoulos (1996). A Microscopic Traffic Simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies* 4 (3), pp. 113–129. DOI: 10.1016/S0968-090X(96)00006-X.

Yao, H., X. Tang, H. Wei, G. Zheng, and Z. Li (2019). Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. Vol. 33. Honolulu, HI: AAAI Press, pp. 5668–5675. DOI: 10.1609/aaai.v33i01.33015668.

Yao, H., F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, D. Chuxing, and Z. Li (2018). Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence.* New Orleans, LA: AAAI Press.

Zhang, J., Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li (2018). Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence* 259, pp. 147–166. DOI: 10.1016/j.artint.2018.03.002.

Zwick, F. and K. W. Axhausen (2020). Analysis of ridepooling strategies with MATSim. DOI: 10.3929/ETHZ-B-000420103.