Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

Doctoral Thesis

David Fiedler

Large-scale Mobility-on-demand: Simulation Studies and Optimization

Ph.D. programme & Branch of study:

(P2612)	Electrical Engineering and Information Technology
(2612V025)	Information Science and Computer Engineering

Supervisor:

prof. Dr. Michal Pěchouček, MSc.

Supervisor-Specialist:

bc. Michal Čáp, MSc., Ph.D.
doc. Ing. Michal Jakob, Ph.D.

February 2024

Acknowledgments

I would like to thank all my co-authors, colleagues and mentors who supported me during the research presented in this thesis. Special thanks belong to my supervisor prof. Michal Pěchouček, and my supervisors specialists Michal Čáp and Michal Jakob for their support and guidance. Next, I would like to thank Jiří Vokřínek, who supported and guided me in completing my research and writing this thesis. Furthermore, I would like to thank my colleagues from the smart mobility group, namely Martin Schaefer, Jan Mrkos, and Marek Cuchý for their support and inspiring discussions. Also, I would like to thank Fabio Vito Difonzo for his steadfast support during the years when we worked together on the plan chaining problem. I would also like to thank prof. Javier Alonso-Mora for letting me spend an inspiring research internship at the Delft University of Technology. Finally, I would like to thank Karel Horák for the very nice LaTeX template that I used for this thesis.

Abstract

Mobility-on-demand (MoD) systems are systems that provide the transportation of passengers on-demand instead of using a fixed schedule. Examples of such systems are transportation network companies (TNCs) like Uber or Lyft, conventional taxi services, or future systems of autonomous self-driving cars being developed by companies such as Waymo, Cruise, or Motional. So far, the most studied MoD systems are small MoD systems focused on servicing people with disabilities or the elderly, these have been studied for several decades, mostly in the field of operational research. In this thesis, we focus on large-scale MoD systems operated by TNCs, which are a relatively new research area. These systems, while formally similar to the small MoD systems, have different characteristics and require different methodologies and algorithms for their study and optimization.

In this thesis, we identified several research gaps in the field of large-scale MoD systems and developed new methodologies and algorithms to address them. First, there are no standardized benchmark instances for large-scale studies available in the literature. We researched the existing benchmark instances and combined the acquired knowledge with our previous experience with simulations of MoD systems. Based on that, we developed a methodology for creating instances from historical travel data and created a set of large-scale instances for three areas: New York City, Chicago, and Washington D.C.

Second, there are several research questions about the operation of MoD that were still unanswered when we started our research. We developed a new simulation methodology that allows for performing large-scale simulation studies of MoD. With that methodology, we deliver answers to three research questions about the operation of MoD: a) what is the impact of replacing private cars with MoD, b) how can we benefit from ridesharing, i.e., by transporting multiple passengers in one vehicle simultaneously, and c) how much we can benefit from dispatching the vehicles optimally. We found out that a) the MoD deployment can generate a significant increase in the total distance driven, despite reducing the fleet size dramatically, b) ridesharing reduces the total distance driven significantly, far below the travel distance of private cars, and c) we can significantly improve the performance further by dispatching the vehicles optimally.

Finally, no algorithm for the optimal dispatching of vehicles in MoD systems is scalable to all practical instances. We have identified a weak point of the optimal method: it fails on instances with a long time horizon, i.e., with travel requests spread over a long time. Based on this finding, we developed a new algorithm that splits such instances into sub-instances covering a shorter time horizon, solves them optimally, and then chains the vehicle plans together. We formulated the plan chaining problem with time windows, developed a method to solve it and proved that it is optimal, and performed a computational study that compares the new dispatching method with the optimal dispatching method, a construction heuristic, and a metaheuristic. For instances that cannot be solved optimally, our method delivered the best solutions among the evaluated methods in the majority of the cases.

Keywords

Mobility-on-demand, simulation, ridesharing, DARP, plan chaining

Abstrakt

Systémy dopravy na vyžádání (anglicky Mobility-on-demand, MoD) jsou systémy, které zajišťují přepravu cestujících na vyžádání namísto použití pevného jízdního řádu. Příkladem takových systémů jsou společnosti jako je Uber nebo Lyft, běžné taxislužby nebo budoucí systémy autonomních samořídících vozidel, které vyvíjejí společnosti jako Waymo, Cruise nebo Motional. Dosud nejvíce studovanými MoD systémy jsou malé MoD systémy zaměřené na obsluhu osob se zdravotním postižením nebo starších osob, ty jsou studovány již několik desetiletí, většinou v oblasti operační analýzy. V této práci se zaměřujeme na MoD systémy velkého rozsahu, které jsou relativně novou oblastí výzkumu. Tyto systémy jsou sice formálně podobné malým MoD systémům, ale mají odlišné charakteristiky a vyžadují odlišné metody a algoritmy pro jejich studium a optimalizaci.

V této práci jsme identifikovali několik mezer ve výzkumu rozsáhlých MoD systémů a vyvinuli nové metody a algoritmy pro jejich zacelení. Za prvé, v existující literatuře nejsou k dispozici žádné standardizované referenční instance pro studie velkého rozsahu. Prozkoumali jsme tedy existující instance a získané poznatky jsme zkombinovali s našimi předchozími zkušenostmi se simulacemi MoD systémů. Na základě toho jsme vyvinuli metodiku pro vytváření instancí z historických mobilitních dat a vytvořili soubor instancí velkého rozsahu pro tři oblasti: New York, Chicago a Washington D.C.

Za druhé, existuje několik výzkumných otázek týkajících se fungování MoD, které byly v době zahájení našeho výzkumu stále nezodpovězené. Vyvinuli jsme proto novou metodiku simulace, která umožňuje provádět rozsáhlé simulační studie MoD. Díky této metodice přinášíme odpovědi na tři výzkumné otázky týkající se provozu MoD: a) jaký by byl dopad nahrazení osobních automobilů MoD, b) jak můžeme těžit ze sdílení jízd, tj. z přepravy více cestujících v jednom vozidle současně, a c) jak velká je výhoda optimálního dispečinku vozidel. Zjistili jsme, že a) nasazení MoD může přinést výrazné zvýšení celkové ujeté vzdálenosti, a to i přes výrazné snížení velikosti vozového parku, b) sdílení jízd výrazně snižuje celkovou ujetou vzdálenost, a to hluboko pod hodnotu v případě použití osobních automobilů, a c) optimálním dispečinkem vozidel můžeme effektivitu dále výrazně zvýšit.

Nakonec jsme řešili problém, že žádný algoritmus pro optimální dispečink vozidel v MoD systémech není škálovatelný na všechny realistické instance. Identifikovali jsme tedy slabé místo optimální metody: selhává na instancích s dlouhým časovým horizontem, tj. s požadavky na jízdu rozloženými do dlouhého časového období. Na základě tohoto zjištění jsme vyvinuli nový algoritmus, který takové instance rozdělí na menší instance pokrývající kratší časový horizont, optimálně je vyřeší a poté plány vozidel zřetězí. Formulovali jsme problém řetězení plánů s časovými okny, vyvinuli jsme metodu pro řetězení plánů a dokázali jsme, že je optimální, a provedli jsme výpočetní studii, která porovnává novou metodu pro dispečink vozidel s optimálním metodou, konstrukční heuristikou a metaheuristikou. U případů, které nelze řešit optimálně, poskytla naše metoda ve většině případů lepší řešení než ostatní testované metody.

Klíčová slova Poptávková doprava, simulace, sdílené jízdy, DARP, řetězení plánů

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published work of others has been acknowledged in the text, and a list of references is given.

In February 2024

Contents

CHAPTER 1

Introduction

In densely populated cities, private cars are considered an unsustainable mode of transportation. Typically, parking capacity is insufficient to accommodate all private vehicles, while the parking spaces occupy a significant portion of the urban space. According to the 2010 estimation by Davis et al. (2010), almost 5 % of the urban area in the Upper Great Lakes region of the United States was occupied by parking spaces. Later, Shoup (2014) analyzed the impact of minimum parking requirements and found that they reduce the average number of units in apartment buildings in Los Angeles by 13 %. On top of that, the existing parking spaces are clearly insufficient, as the drivers spend a significant amount of time searching for parking spaces. An average time among ten large US cities was estimated to be 17 hours per year (INRIX, 2017). Similarly, the current road capacity is insufficient for all private vehicles, as we can see from the congestion levels in metropolitan areas. According to the 2017 study among US cities, drivers lost on average 41 hours of their time in congestion (Cookson, 2018), and the recent study of the year 2022 increased this number to 51 hours (Bob Pishue, 2023). And all this is happening while the government spends more than $100 billion on road infrastructure per year (International Transport Forum, 2024), part of which is spent on the expansion of the road capacity. Also, apart from the cost, we have to consider that the road capacity is hard to expand in the cities due to the lack of urban space. Finally, cars are also a significant source of air pollution, which has proven to be a significant factor in the development of various diseases and increased mortality (Boogaard et al., 2022). Although we have observed these problems for many years, the situation does not seem to be improving. In fact, it is expected to worsen in the future, as the number of people living in cities is expected to grow (68 % of the world's population is expected to live in urban areas by 2050 (United Nations, 2018)), and the number of vehicles per capita is expected to grow as well. Only from 2016 to 2020, the number of vehicles per capita in the EU grew by more than 7 % (*Vehicles in Use Europe 2022*, 2022), and in India by as much as almost 36 % (*Road Transport Year Book*, 2023).

One of the proposed solutions to address these problems is the deployment of metropolitan mobility-on-demand (MoD) systems providing an alternative to traveling in a private vehicle designed to be as comfortable as traveling in a private car but with smaller parking capacity and road capacity requirements (Alonso-Mora et al., 2017; Čáp & Alonso-Mora, 2018; Miller & How, 2017; Spieser et al., 2014). These MoD systems consist of a fleet of shared passenger vehicles that jointly serve the travel requests of the system's users. For each incoming travel request, the MoD system assigns the request to one of the vehicles and alters its route such that the passenger is picked up and transported to the drop-off location. MoD systems sequentially transport multiple passengers with the same vehicle, so they can serve the existing transportation demand with a smaller, highly-utilized vehicle fleet and thus, significantly reduce the need for urban parking space. To further improve the system's efficiency, the provider can implement *ridesharing*, where multiple passengers can be transported in one vehicle simultaneously (Alonso-Mora et al., 2017). Efficient ridesharing increases vehicle occupancy, which consequently reduces the required fleet size and total distance driven by the vehicle fleet, resulting in ecological and economic benefits.

However, these systems do not exist only in the visions presented in the literature but are available for a long time all around the world. An example of such service is the classical taxi service, emerging on the streets of London at the beginning of the 17th century in the form of horse-drawn carriages (Sir Walter Gilbey, 1903) (see Figure 1.2a). Later, these carriages were replaced by motorized vehicles and other modes of transportation, such as planes, also started to be used for on-demand mobility (Gower & Spicer, 1932). In the latest decade, we can see a rise of a new form of MoD operated by large companies such as Uber or Lyft (Figure 1.2b). These companies leverage the latest digital technologies to provide more efficient dispatching of vehicles, more transparent and accurate pricing, and faster and more convenient ordering of the service. Although the recent coronavirus pandemic has been a significant setback for these companies, the MoD is again on the rise, as we can see from the number of MoD trips per day in New York City displayed in Figure 1.1. In the future, MoD

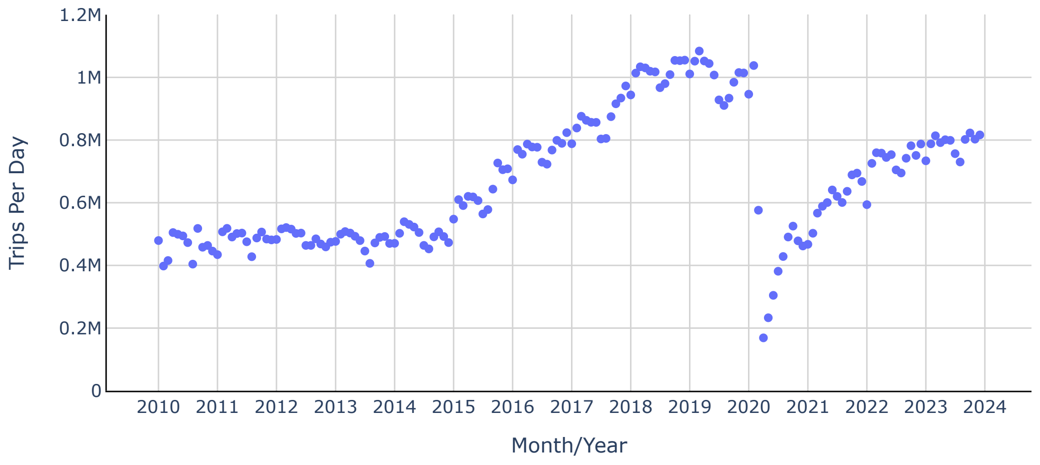


Figure 1.1: Number of MoD trips per day in New York City (Data source: ("Aggregated Reports - TLC", n.d.)).

(a) Early horse-driven carriage (Sir Walter Gilbey, 1903).

(b) Modern digital MoD (front) and classical taxi service (back) operating in NYC (Annie McDonough, 2019)

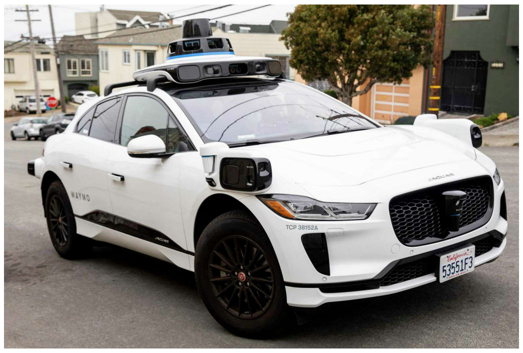

(c) Autonomous MoD operated in San Francisco. (Cano, 2023)

Figure 1.2: History of the vehicular MoD.

is expected to be one of the first applications of autonomous vehicles (Aditya Ambadipudi et al., 2017). In fact, these systems of autonomous MoD, typically called AMoD or robo-taxi, are already being put into operation in some cities. The first such system was launched in 2016 in Singapore by the company nuTonomy (now Motional) (Bhagat, 2016). The system was free of charge and the safety driver was present in the vehicle at all times. However, in 2023, the companies Waymo and Cruise received approval to offer autonomous MoD services in San Francisco without a safety driver at all hours of the day and charge a fare for the service (*Public Utilities Commission of the State of California*, 2023a; *Public Utilities Commission of the State of California*, 2023b). As of 7th August 2023, these two companies operated more than 550 AMoD vehicles in San Francisco (Cano, 2023) (an example Waymo vehicle is in Figure 1.2c).

With the growing share of modern digital MoD systems operated by large companies, there is an unprecedented potential to increase the efficiency of MoD. However, to measure and maximize the benefits of such systems, it is necessary to understand their behavior and performance and to develop new algorithms and methodologies to optimize them. As this field is relatively new, there are still many open questions and there is a significant potential for optimization. In this thesis, we summarize our research on MoD systems. We developed a set of benchmark instances for MoD systems, evaluated them in a set of case studies, and published them together with the methodology for creating them. Using our methodology for simulation studies of MoD systems, we have been able to answer several research questions about the operation of MoD systems. Based on the findings, we have developed a new algorithm that can be used to improve the MoD systems' performance both in terms of total distance driven and fleet size. Mostly, this thesis is focused on the operation of MoD systems, i.e., on the assignment of travel requests to vehicles and the routing of vehicles in the road network. To get a broader picture of the research challenges related to MoD systems, refer to the review by Zardini et al. (2022).

1.1 Research Goals

The goals of the research summarized in this thesis can be divided into two main groups: a) to answer research questions about MoD systems operation using simulation studies and b) To optimize MoD systems using new algorithms and methodologies. Each particular research goal is listed in Table 1.1 together with the corresponding article and the chapter where the research goal is addressed. In the remainder of this section, we provide a brief overview of each research goal.

Code	Group	Goal	Article	Chap.
RG1	a, b	Create and publish realistic large-scale MoD benchmark instances	(Fiedler & Mrkos, 2023a)	3
RG2	a	Analyze the impact of empty trips in MoD systems	(Fiedler et al., 2017)	4
RG3	a	Quantify the benefits of ridesharing in MoD systems	(Fiedler et al., 2018)	4
RG4	a	Quantify the benefits of optimal ridesharing	(Fiedler, Čertický, Alonso-Mora, et al., 2022)	4
RG5	b	Optimize vehicle dispatching in MoD	(Fiedler et al., 2024)[1]	5

[1] Manuscript titled "Optimal Chaining of Vehicle Plans with Time Windows" submitted to the journal *Transportation Research Part C: Emerging Technologies*.

Table 1.1: Research goals with corresponding articles and chapters where they are addressed. The research goals are divided into two groups: a) answering research questions with simulation studies and b) optimizing MoD systems.

RG1: Create and publish realistic large-scale MoD benchmark instances: As for most domains, we need to have a set of problem instances (input data) to perform the research on analysis, or algorithms for MoD systems. During the research, we have discovered that the existing instances are either small-scale, uniformly generated with characteristics that do not correspond to the current large-scale MoD systems (operational research instances (Cordeau, 2006; Cordeau & Laporte, 2003)), or they are not publicly available and therefore, the research is obstructed by the inability to compare or reproduce the results. Moreover, in the latter case, a significant proportion of the instances are located in the same area, Manhattan, New York (Alonso-Mora et al., 2017; Beirigo et al., 2022; Fiedler, Čertický, Alonso-Mora, et al., 2022; Haliem et al., 2021; Santi et al., 2014; Seo & Asakura, 2022; Thangaraj et al., 2017; Vazifeh et al., 2018; Wallar et al., 2019b; W. Zhang et al., 2023), which can lead to biased results. To address these issues, we decided to develop a methodology for creating large-scale MoD instances based on historical travel data and publish this methodology together with a set of instances located in various areas.

RG2: Analyze the impact of empty trips in MoD systems: In most cities, MoD systems are already available, operated by various companies. However, they typically form only a small fraction of the total transportation. In our research, we want to use simulation studies to analyze the impact of large-scale MoD systems deployment, specifically, we analyze a scenario where all the private cars are replaced by MoD. When private cars are replaced by MoD, the total number of vehicles needed is much lower(Bischoff & Maciejewski, 2016; Fagnant & Kockelman, 2014; Spieser et al., 2014), and we want to quantify the benefits of this replacement. However, the MoD systems also generate empty trips between individual passengers and also to compensate for unbalanced travel demand. Therefore, apart from quantifying the benefits of reducing the number of vehicles like in (Spieser et al., 2014) and (Bischoff & Maciejewski, 2016), we also want to analyze the negative effects of empty trips: a) on total vehicular traffic, b) on traffic congestion and c) on passenger delay.

RG3: Quantify the benefits of ridesharing in MoD systems: To compensate for the empty trips in MoD systems, we can implement ridesharing, i.e., transport multiple passengers in one vehicle simultaneously. Nevertheless, it is not clear how much ridesharing reduces the total travel distance and the number of vehicles needed. This research goal is therefore to extend the simulation methodology from RG2 with the support of dynamic ridesharing and use it to quantify the benefits of ridesharing in large-scale MoD systems. Unlike Alonso-Mora et al. (2017) who analyze the scenario of replacing taxis in Manhattan, we want to analyze a scenario of replacing all private cars with MoD systems. Also, apart from the travel distance reduction, we want to analyze the impacts on the fleet size, traffic congestion, and the delay of passengers.

RG4: Quantify the benefits of optimal ridesharing: Most of the existing ridesharing studies use heuristic methods for vehicle dispatching. A logical question is how much we can benefit from using optimal methods instead. In this research goal, we want to extend the simulation methodology from RG3 with optimal an ridesharing algorithm and use it to quantify the benefits of optimal ridesharing in large-scale MoD systems. In addition to the study by Alonso-Mora et al. (2017), we want to test an optimal method without any resource constraints or relaxation, providing an upper bound on the benefits of ridesharing. On the other hand, in contrast to Čáp and Alonso-Mora (2018), who use the optimal variant of VGA as well, we want to evaluate the method on a much larger scale and on realistic instances. Finally, we, want to compare an optimal method to both resource-constrained variants of VGA and the insertion heuristic, efficiently quantifying the benefits of optimal ridesharing and also its cost in terms of computational time. This comparison, still missing in the literature, could provide insights into the trade-offs between the solution quality and the computational time.

RG5: Optimize vehicle dispatching in MoD: Our ultimate goal is to use our insights on MoD systems to make progress in the optimization of vehicle dispatching. In our simulation studies, we have identified instances that are impossible to solve optimally using the existing methods, and, at the same time, have low-quality solutions when using relaxed variants of the methods. Our goal is to develop a new heuristics algorithm that can provide high-quality solutions for these instances while being able to solve them in a reasonable time.

1.2 Why to Read This Thesis

As most of the key findings present in this thesis have been already published as separate articles, it is appropriate to explain what it can offer to the reader. First, the thesis provides extra content not present in any of the published articles. The Chapter 5 dedicated to MoD optimization is based on the article that is currently under review. Next, we provide an extensive terminology section that helps to distinguish between similar terms related to MoD in Chapter 2. Finally, as the articles are limited in length, the thesis provides extra content even in the chapters that are based on the published articles. This includes an extended related work (e.g., Section 3.1.3), more details about the methods (e.g., Section 4.2.3), or analysis of algorhitms (e.g., Section 2.3.1, 2.3.2, or 5.3.1).

Second, this thesis puts the results of the articles into a bigger context. Based on the self-contained articles, the thesis provides a comprehensive body of knowledge about simulations and optimization of mobility-on-demand systems. It unifies the notation and terminology used in the articles, and it also explains the terminology choices and alternatives.

1.3 Outline

The content of this thesis is organized in the logical order of the research to be easily accessible to the reader, not in the chronological order of the publication of the articles. We start with the terminology, notation, and problem formulations in Chapter 2. Then, we discuss the possibilities of obtaining input data for modeling MoD systems and present a methodology for creating large-scale instances based on historical travel data (Chapter 3). In Chapter 4, we describe our simulation methodology and present the results of the simulations which answer the research questions about MoD systems operation. Finally, in Chapter 5, we present a new algorithm for optimizing dispatching in MoD systems with ridesharing which is inspired by the results of the simulation studies.

CHAPTER 2

Preliminaries

In this chapter, we provide the background material on which the rest of this thesis is built. Specifically, we provide the terminology, notation, and problem formulations that are common for the following chapters. Further extensions of the introduced concepts specific to each chapter are discussed in the respective chapters.

2.1 Background and Terminology

In this section, we lay down the terminology used in the rest of this thesis. Besides the introduction of each term, special care is taken to explain and justify the choice of the term,

---

This chapter is based on the journal article:

Fiedler, D., Čertický, M., Alonso-Mora, J., Pěchouček, M., & Čáp, M. (2022). Large-scale online ridesharing: The effect of assignment optimality on system performance. *Journal of Intelligent Transportation Systems*, *0*(0), 1–22. https://doi.org/10.1080/15472450.2022.2121651

the following conference papers:

(Fiedler et al., 2017) Fiedler, D., Čáp, M., & Čertický, M. (2017). Impact of mobility-on-demand on traffic congestion: Simulation-based study. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC.2017.8317830

(Fiedler et al., 2018) Fiedler, D., Čertický, M., Alonso-Mora, J., & Čáp, M. (2018). The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1173–1178. https://doi.org/10.1109/ITSC.2018.8569451

(Fiedler & Mrkos, 2023b) Fiedler, D., & Mrkos, J. (2023b). Large-scale Ridesharing DARP Instances Based on Real Travel Demand, 2750–2757. https://doi.org/10.1109/ITSC57777.2023.10422146

and the manuscript titled “Optimal Chaining of Vehicle Plans with Time Windows” submitted to the journal *Transportation Research Part C: Emerging Technologies*. Preprint:

Fiedler, D., Difonzo, F. V., & Mrkos, J. (2024, February 24). *Optimal Chaining of Vehicle Plans with Time Windows*. arXiv: 2401.02873 `[cs, math]`. https://doi.org/10.48550/arXiv.2401.02873

For all author’s publications together with the citation counts and research contributions, see Appendix A.

as the terminology in the field of MoD is not consistent and sometimes even ambiguous.

2.1.1 Mobility-on-demand Systems

The term *mobility-on-demand* (MoD) emerged at the end of the first decade of the 21st century in the documents of MIT Smart Cities group[1]. At first, the term referred to vehicle sharing systems, where the vehicles (car, bicycle,...) are shared among multiple users who can pick up and drop off the vehicle at MoD stations without the need to reserve it in advance (Mitchell et al., 2010). However, later, the meaning of the term has been extended to include also systems operated by *transportation network companies* (TNCs) such as Uber or Lyft (Vazifeh et al., 2018), or for future systems of autonomous self-driving cars (AMoD) (Beiker, 2016). Nowadays, the term MoD is used to refer to any system that provides on-demand transportation service to its users (Susan Shaheen & Adam Cohen, 2020).

2.1.2 Carsharing, Ridesharing, Carpooling, and Ride-pooling

The key feature of MoD systems is that a single vehicle can transport multiple passengers sequentially. This is true both for vehicles operated by *transportation network companies* like Uber or Lyft and for *vehicle sharing systems* (Nair et al., 2013) functioning essentially as short-term rental services (In the case of cars, this concept is called *carsharing* (Shaheen & Cohen, 2013)).

Reaching vehicle occupancy greater than one person per vehicle is the single most important component of the future MoD systems if we want to optimize the efficiency (Fiedler et al., 2018). Unfortunately, this concept of transporting multiple passengers in one vehicle simultaneously has inconsistent and ambiguous terminology. The most used term to describe it is *ridesharing*. In the most general sense, the term ridesharing describes a situation, system, or setting, where a part of a trip (in a vehicle) is shared between multiple passengers (Alonso-Mora et al., 2017; Fiedler et al., 2018; S. Ma et al., 2015). This is the meaning we use in this thesis.

However, in older articles, ridesharing usually translates to sharing the ride with someone with an equal or similar start/end location, for example, while commuting (Hartgen, 1977). Today, this is usually called *carpooling* (Ferguson, 1997; J. Li et al., 2007). To complicate it even more, the term carpooling is also sometimes used in a broader sense, to describe any form of shared ride (i.e., interchangeably with the term ridesharing). However, in this thesis, we will never use the term carpooling in this broader sense.

Finally, another term that is used interchangeably with ridesharing as described above is *ride-pooling* (Engelhardt et al., 2019; Santi et al., 2014). Nevertheless, to avoid confusion, we will refrain from using the term ride-pooling in this thesis.

[1]https://smartcities.media.mit.edu

2.1.3 Transportation Network Companies and Ridehailing

Most of the on-demand transportation is realized by transportation network companies like Uber and Lyft. These companies evolved as a more up-to-date alternative to well-established taxi services. They offer a centralized service through users' cell phones app instead of waiting to be hailed on the streets, or requested by a phone call. Because the centralization, the service can be optimized and the price is usually known before the ride itself. TNCc are sometimes incorrectly referred to as ridesharing companies, despite the users of these services do not share rides (with some exceptions like Uber Pool).

The traditional service provided by TNCs, i.e., servicing each user with a dedicated vehicle, is called *ride-hailing* (Banning-Lover, 2021). Sometimes even ridesharing service (as described in the previous section) is called ride-hailing if operated by TNCs. However, we will refer to such service as ridesharing, even if it is operated by a TNC.

2.1.4 Dial-a-ride Problem and Its Relation to MoD

A long time before the term MoD and the other terms explained in this chapter were established, the concept of transporting passengers with fleets of vehicles was known as a *dial-a-ride (DAR) service* or a *dial-a-ride system* (Guenther & Givens, 1970; National Research Council (U S.), Highway Research Board, 1971). At that time, the considered fleets were much smaller and the system was mostly designed to transport elderly or people with disabilities (Toth & Vigo, 2014). However, the main principle of dial-a-ride systems is the same as the principle of MoD systems: to offer a platform for requesting transportation on-demand.

Although the terms dial-a-ride service and dial-a-ride system have become obsolete due to the technology change used to order the service, they left their legacy in the form of another term: *dial-a-ride problem* (DARP). The DARP is a problem of finding optimal vehicle plans for a fleet of vehicles transporting passengers between their pickup and drop-off locations (Toth & Vigo, 2014).

So far, we have described the *static* variant of the DARP. However, there exists also *dynamic (online)* variant. In dynamic DARP, requests arrive over time and the system has to react to them immediately or within a short time horizon. This means that instead of computing the solution for a long time period (day), we compute the solution only for a short time period (seconds or minutes) iteratively, solving a new DARP instance for each time period. Formally, these DARPs are almost the same as the static DARP but there are differences in the problem characteristics. Most importantly, the requests in one dynamic DARP instance have similar time constraints, usually much tighter than in static DARP instances.

2.1.5 Fleet-sizing and Rebalancing

One of the key problems regarding MoD systems is to determine the minimal vehicle fleet able to serve all travel requests: the *fleet sizing* problem. In the ideal MoD system design, the fleet size and the operating policy of the system are obtained by solving a multiobjective optimization problem, where the objectives are the total distance driven, fleet size, and the passengers' discomfort. However, in the vast majority of the research works on MoD systems, these problems are solved separately, i.e., the fleet size is determined first and then the operating policy is designed for the given fleet size.

Another challenge of MoD systems is the disbalance of the transportation demand. The demand is usually not uniformly distributed in space and time, which results in the imbalance of the number of vehicles in different parts of the city. For instance, during the morning peak, the vehicles are typically requested for pickup in residential areas, but they subsequently end up in business districts. As a result, the stock of vehicles in residential areas is shrinking, while unused vehicles are accumulating in business areas. To tackle this problem, the MoD system provider can use *rebalancing*, i.e., the relocation of vehicles from areas with a surplus of vehicles to areas with a shortage of vehicles. (Spieser et al., 2014).

2.2 Notation and Problem Formulations

In this section, we provide a unified notation for this thesis and we also formulate two most important problems in MoD: the dial-a-ride problem and the fleet-sizing problem. Other more specific notation and problem formulations are introduced through the rest of the thesis as needed.

2.2.1 MoD Notation

Transportation Demand

The key input to any problem related to MoD systems is the transportation *demand*. The demand D is a set of travel requests representing users that want to travel from some *origin* point to the desired *destination*, respecting some time constraints.

The most general form of these time constraints are the so-called *time windows* that restrict both minimum and maximum times for origin and for destination. Each request is thus defined as a pair (o, d) where o (origin) and d (destination) are triples $(l, t^{\min}, t^{\max})$ where l is the *location* and $t^{\min}$ and $t^{\max}$ are minimum and maximum pickup (or drop off) times.

In online scenarios, however, we usually assume that there is no restriction on the minimum drop-off time (users want to be transported as soon as possible). Also, instead of separate maximum times for origin and destination, we assume that each user is willing to delay his travel by the same max delay constant $\delta_{\max}$. As a result, the request can be simplified as a triple (o, d, t), where o and d are origin and destination locations, and t is the desired pickup time of the request. In this thesis, we will use this simplified notation. Note that the delay δ can be caused by two factors: late pickup or detour while the request is already on board. The total delay is computed as:

$$\delta = t'_d - f_{\mathrm{tt}}(o,\ d) - t_o, \tag{2.1}$$

where t'_d is the actual drop-off time of the request.

Locations and Time

In the previous section, we defined a request as a triple (o, d, t), where o and d are locations and t is a time. However, the domain for locations and time is not specified yet. In fact, we will not specify the format for locations at all, as it is not important for the problems discussed in this thesis. We only use locations to measure the travel time or travel cost between two locations and for that, the location can have any form as long as the travel time/cost function is defined.

We assume that travel times between any two locations a and b are known in the form of a travel time function $f_{\mathrm{tt}}(a,\ b)$. Note that the function parameter order matters here, as in real road networks, it frequently holds that $f_{\mathrm{tt}}(a,\ b) \neq f_{\mathrm{tt}}(b,\ a)$. Analogously, we define the travel cost function $f_{\mathrm{tc}}(a,\ b)$, which represents the travel cost. Note that if we optimize for minimum travel time, we can set $f_{tc} = f_{tt}$.

Similarly to locations, we do not specify the unit of time. We only assume that each time property (pickup time, travel time, etc.) is specified using the same unit and that the codomain of the travel time function $f_{\mathrm{tt}}(a,\ b)$ represents the same unit of time.

Vehicles and Vehicle Plans

Every request in an MoD system is served by one of the available vehicles; we denote the sequence of all available vehicles as v. In traditional DARP problems, vehicles are parked in single or multiple depots (Toth & Vigo, 2014), but in general, vehicles can start at any location. Another important parameter for vehicles is capacity: the number of users that can travel in the same vehicle simultaneously. Therefore, we define a vehicle as a pair (l, c), where l is the initial location of the vehicle, and c is its capacity.

The plan of a vehicle is represented as a sequence of locations $p = l_1, l_2, \ldots l_{|p|}$, where each location is either an origin location o_r, or a destination location d_r of request r that is scheduled to be serviced by the plan. A vehicle plan is *valid* only if a) for each request r serviced by the plan, the plan contains the origin location o_r, and the destination location d_r exactly once, and b) for each request r, o_r appears before d_r in the plan. An important property of a valid vehicle plan is its *feasibility.* A vehicle plan is *feasible* if it respects the time constraints of all requests and the capacity constraint of the vehicle. The (operating) cost of a vehicle plan p denoted as $f_{\mathrm{tc}}(p)$ is defined as the sum of travel costs between consecutive locations in the plan $(l_1, \cdots, l_{|p|})$, starting with the vehicle location l_v:

$$f_{\mathrm{tc}}(p) = f_{\mathrm{tc}}(l_v,\ l_1) + \sum_{i=1}^{|p|-1} f_{\mathrm{tc}}(l_i,\ l_{i+1}). \tag{2.2}$$

There can be multiple valid plans for a single vehicle. We denote the set of all valid plans for a vehicle v as P_v. This set is never empty as the *empty plan* with no requests is by definition valid for any vehicle.

To explain some methods and algorithms, it is also convenient to define a *system plan* as a set of one plan for each vehicle. A system plan π is each set of plans for which the following holds:

$$\forall v \in V : \exists! p \in P_v, p \in \pi. \tag{2.3}$$

An extended formulation for vehicles is required for the dynamic DARP problem, as we need to consider the state resulting from the previous DARP instance. Here, we extend the vehicle definition with a property R which is the set of requests that are currently being transported by the vehicle.

2.2.2 DARP Instances

With the introduced notation we can finally define the problem instance for the DARP. A DARP instance is composed of:

- a set of requests D,
- a sequence of vehicles v,
- a travel time model whose most basic example is the travel time function function $f_{\mathrm{tt}}(a,\ b)$,
- and a travel cost model whose most basic example is the travel cost function $f_{\mathrm{tc}}(a,\ b)$.

Various instance parameters can be used to further constrain the problem, so for each set of instances, we need to specify which parameters are used. Typical some subset of the following parameters is used:

- *maximum allowed delay* $\delta_{\max}$, if not specified individually for each request,
- *service time* at each request, which represents the time needed to pick up and drop off the MoD user,
- requirement for vehicles to return to the initial location (depot) after the last request,
- the *max ride time*, i.e., maximum allowed travel time for request,
- and the *max route time*, the maximum allowed travel time for any vehicle plan.

The max ride time limiting the travel time of a request is usually needed for instances with loose time constraints, to prevent a flexible passenger from spending too much time in the vehicle. The max route time represents the maximum time a driver can work without pause imposed by contractor law.

Apart from the instance definition, each instance has some important properties that can be directly derived from its components. The first of such properties is the *time horizon*, denoted as τ. This interval represents either the allowed operating time of the vehicles or the time interval between the earliest pickup time and the latest drop-off time in case the vehicle operating time is not limited. The second property is the *time window length*, denoted as ω. It represents the time flexibility of the requests' pickup and drop-off times. If the $\delta_{\max}$ is specified for the whole instance, the window length is equal to it. Otherwise, we know the interval of the possible values of ω and we can also compute the average window length $\bar{\omega}$. As we have explained, both these two properties can be directly derived from the instance definition. The reason why we mention them separately is that they are the most significant indicators of the instance combinatorial complexity. Therefore, when describing or comparing some instances, it is usually sufficient to list these properties together with the number of requests and vehicles, instead of specifying the instance definition.

2.2.3 Dial-a-ride Problem

The DARP problem consists of assigning a vehicle to each request and computing feasible plans for all vehicles. Note that in order to receive optimal or high-quality sub-optimal solutions, these two aspects need to be solved together. Unfortunately, this is not always clear in the MoD literature, where DARP is frequently referred to as *request-vehicle assignment* or *trip-vehicle assignment*, despite the process result in vehicle plans, not just in matches between passengers and vehicles (Alonso-Mora et al., 2017; Čáp & Alonso-Mora, 2018).

There are multiple criteria that can be optimized. Usually, we optimize the operation cost, service quality, or both. For simplicity, these two criteria are usually reduced to the total travel distance and the total delay, respectively.

To state the above more formally: when solving DARP, we try to find the optimal *system plan* (set of vehicle plans) such that: 1) every request is served, 2) all plans are feasible, and 3) the total cost is minimized. Below, we define the DARP for the case of optimizing for the minimal total travel distance.

Problem 2.1. Find the set of feasible plans P that minimize

$$\sum_{p \in P} f_{\mathrm{tc}}(p) \tag{2.4}$$

subject to

$$\exists! p \in P_v, p \in P \quad \forall v \in V \tag{2.5}$$

$$\exists! p \in P, o_r \in p \quad \forall r \in D \tag{2.6}$$

Note that the above problem formulation is not the classical DARP formulation. Typically, DARP is formulated as a mixed-integer program (MIP). However, for this thesis, the above formulation is sufficient, as we do not solve DARP directly using MIP solvers. For the classical MIP formulation, refer to Cordeau (2006) or Ropke and Cordeau (2009).

Finally, let us discuss the complexity of the DARP. A special case of the DARP with unbounded time constraints is the capacitated vehicle routing problem (CVRP) (Toth & Vigo, 2014). If we consider even more special cases with unbounded capacity, the problem becomes the vehicle routing problem, and by considering only one-vehicle instances, it becomes the traveling salesman problem (TSP), which is a generalization of the Hamiltonian cycle problem. As the Hamiltonian cycle problem belongs among the 21 NP-complete problems introduced by Karp (1972), we can immediately prove, by restriction, that the DARP is NP-hard. Since the DARP has time and capacity constraints, it is interesting to look at the problem of finding any solutions to the DARP. In fact, an optimal solution can be also the only solution, or there can be no solution at all. Therefore, even determining the feasibility of a DARP instance is NP-hard, as proved by Savelsbergh (1985). This has important implications: even heuristic methods for solving DARP can be computationally expensive or not capable of finding a solution for solvable instances.

2.2.4 Fleet-sizing Problem

The fleet sizing problem is a problem of determining the fleet size $|V|$. As we mentioned in Section 2.1.5, the fleet-sizing problem is usually solved separately from the MoD system operation optimization (DARP problem). Therefore, we define it here as a single-objective optimization problem, where the objective is to minimize the fleet such that the MoD service level is acceptable. There are multiple possible definitions of the acceptable service level. Here, we define it as the maximum number of unserved requests ϵ when solving DARP for a given demand or a set of demands. In some works, the threshold ϵ is set to zero, i.e., the fleet size is minimized such that all requests are served (Fiedler, Čertický, Alonso-Mora, et al., 2022), while in others, dropped requests are allowed (Alonso-Mora et al., 2017).

2.3 DARP Solution Methods

Through all chapters of this thesis, we use DAPR solution methods for various tasks. In Chapter 3, we use them to determine the fleet size for the given demand (fleet-sizing) and also for the demonstration experiments. In Chapter 4, we simulate various MoD systems, some of which require DARP solutions methods for vehicle dispatching. Finally, in Chapter 5, we use existing DARP solution methods as a building block for new solution methods, and as a baseline for comparison. Therefore, in this section, we present two DARP solution methods that we use the most in this thesis: the Insertion Heuristic and the Vehicle-group Assignment (VGA) method.

2.3.1 Insertion Heuristic

The Insertion Heuristic (IH) is a simple and fast heuristic introduced by Jaw et al. (1986) in 1986. To this day, it is one of the most used heuristics for solving the DARP (Bischoff et al., 2017; Campbell & Savelsbergh, 2004; Fiedler et al., 2018; Kalina et al., 2015). Also, IH is often used as a subcomponent of more sophisticated algorithms. For example, in ridesharing with demand prediction (van Engelen et al., 2018), when integrating ridesharing with public transport (T.-Y. Ma et al., 2019), or as an initial solution generator for metaheuristic methods (Muelas et al., 2013).

The IH logic processes the requests one by one, typically in the order of their desired pickup time (arrival in online DARP). For each request, the IH attempts to insert the request into the plan of some vehicle so that the resulting overall cost of all plans is the lowest. To compute this minimum cost increment, the algorithm tries all vehicles, and for each vehicle, it tries all possible pickup and drop-off insertion points. In other words, the

IH inserts a request into the system plan optimally, the suboptimality comes from the fact that the requests are processed one by one.

The pseudocode of the IH is presented in Algorithm 1. The inputs of the algorithm are the set of requests D and a set of vehicle plans p, one for each vehicle. These initial vehicle plans are empty in static DARP, but in the dynamic DARP, these plans contain the requests that are already being transported by the vehicles. Note that when adding a new request into a plan p_i, the relative ordering of all locations from p_i remains unchanged in the new plan, and therefore, optimality is not guaranteed.

Algorithm 1 Insertion Heuristic

```
Input: set of requests $D$, sequence of empty vehicle plans $p$, one for each vehicle
for $r \in D$ do
    $\delta_c^{\min} \leftarrow \infty$                              ▷ minimum cost increment
    $p^* \leftarrow$ null                                   ▷ best plan
    $i^* \leftarrow 0$                                      ▷ index of the best plan
    for $i \in 1, \ldots, |p|$ do
        for $j \in 1, \ldots, |p_i| + 1$ do
            for $k \in j + 1, \ldots, |p_i| + 2$ do
                $p^{\text{new}} \leftarrow p_i$
                insert $o_r$ to $p^{\text{new}}$ on index $j$ and $d_r$ on index $k$
                if $p^{\text{new}}$ is feasible then
                    $\delta_c \leftarrow f_{\text{tc}}(p^{\text{new}}) - f_{\text{tc}}(p)$
                    if $\delta_c < \delta_c^{\min}$ then
                        $\delta_c^{\min} \leftarrow \delta_c$
                        $p^* \leftarrow p^{\text{new}}$
                        $i^* \leftarrow i$
    if $p^*$ not null then
        $p_{i^*} \leftarrow p^*$
```

In the remainder of this section, we discuss the complexity of the IH. For each request, the IH tries all possible combinations of pickup and drop-off insertion points for each vehicle plan.

Assumption 2.1. A single step in the IH algorithm is to try to insert a pair of pickup/drop-off actions into a plan.

We also assume a realistic ratio of vehicles to requests:

Assumption 2.2. $|V| \leq c \cdot |D|$, where c is constant

With these assumptions, we can show that the asymptotic worst-case complexity of IH is cubic to the number of requests, as stated in the following theorem:

Theorem 2.1. *Under the Assumptions 2.1 and 2.2, the asymptotic worst-case complexity of Algorithm 1 is* $O\left(|D|^3\right)$.

Proof. In IH the requests are processed one by one, and the algorithm is greedy, i.e., previous requests cannot be reordered. The number of possible pairs of pickup/drop-off positions is the sum of possible insertions in two categories: 1) insertion to empty plan, and 2) insertion among already inserted requests. For 1), the number of possible combinations is $|V|$, as a drop-off has to be inserted after the pickup, so there is only one possible pickup/drop-off combination for each plan, and there is exactly one empty plan for each vehicle. For 2), we need to understand that we can only insert a new action after an already inserted action or at the beginning of the plan. Otherwise, we are inserting it into an empty plan, which is already covered in 1). For i-th request, there are $2(i-1)$ possible pickup/drop-off indices (positions) after already inserted actions and one at the beginning of the plan, in total $2(i-1)+1 = 2i-1$ possible indices. The number of ordered pairs of pickup and drop-off indices is therefore:

$$(2i-1)^2 \tag{2.7}$$

However, the drop-off has to be inserted after the pickup, so the index of the drop-off has to be equal to or greater than the index of the pickup. Given n possible indices, this means that for i-th pickup index, there are $n-i+1$ possible dropoff indices. We can now compute the number of possible ordered pairs as a sum:

$$\sum_{i=1}^{n} n-i+1 = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1 = n^2 - \frac{n(n+1)}{2} + n = \frac{n^2+n}{2} \tag{2.8}$$

In our case, $n = 2i-1$, so the number of valid pairs for i-th request is:

$$\frac{(2i-1)^2+2i-1}{2} = \frac{4i^2-4i+1+2i-1}{2} = 2i^2 - i \tag{2.9}$$

Now, we can compute the total number of possible ordered pairs for all requests as a sum over requests:

$$\sum_{i=1}^{|D|} |V| + 2i^2 - i = |V||D| + \sum_{i=1}^{|D|} 2i^2 - \sum_{i=1}^{|D|} i \tag{2.10}$$

$$= |V||D| + \frac{2|D|^3}{3} + |D|^2 + \frac{|D|}{3} - \frac{|D|^2}{2} - \frac{|D|}{2} \tag{2.11}$$

$$= |V||D| + \frac{2|D|^3}{3} + \frac{|D|^2}{2} - \frac{|D|}{6} \tag{2.12}$$

The asymptotic complexity resulting from the above is $O\left(|V||D| + |D|^3\right)$. However, we may simplify it further by applying the assumption 2.2 Given that, the complexity can be rewritten as:

$$O\left(k|D|^2 + |D|^3\right) = O\left(|D|^3\right) \tag{2.13}$$

□

Algorithm 2 VGA method. The sequence γ consists of sets of feasible groups, one set for each vehicle. The optimal system plan is denoted as π^*.

```
Input: set of requests $D$, sequence of vehicles $v$
  $\gamma \leftarrow ($ generate_groups($D$, $v_i$) $)_{i=1}^{|v|}$        ▷ group generation
  $\pi^* \leftarrow$ Solve Problem 2 using $\gamma$        ▷ vehicle-group assignment
```

To conclude the complexity analysis, we wish to highlight two facts that might contradict initial intuition: a) the complexity is one order lower than the number of loops in the Algorithm 1, and b) if the number of vehicles is proportional to the number of requests, the asymptotic complexity is independent of the number of vehicles.

2.3.2 Vehicle-group Assignment Method

The Vehicle-group Assignment (VGA) method is an optimal method for solving the DARP. It was introduced by Alonso-Mora et al. (2017) as a method for solving large-scale online ridesharing problems. Later, it was used by Čáp and Alonso-Mora (2018) for static DARP.

The main principle of the VGA method is first to compute all feasible plans for each vehicle, the *group generation* phase, and then solve the problem of assigning plans to vehicles, the *vehicle-group assignment* phase. The overall pseudocode of the VGA method operation is in Algorithm 2, and the whole procedure is demonstrated by an example in Figure 2.1. First, we use the group generation procedure (the `group_generation` function in Algorithm 2) described in the following section to generate sets of all feasible plans for each vehicle. A sequence of these sets is denoted as γ. Then, we obtain an optimal system plan π^* by solving Problem 2, the vehicle-group assignment, with γ as an input. This is described in Section 2.3.2.

Group Generation

The group generation process consists of generating groups of requests that can be served by a vehicle, and the corresponding vehicle plans. We define a *group* R as a set of requests such that $R \subseteq D$. We say that a group R is *feasible* for vehicle a if there exists a feasible plan for a that can serve all the requests in R. We denote the set of all feasible groups for a as $\gamma_a \subseteq \mathcal{P}(D)$.

The group generation procedure is shown in Algorithm 3. It leverages the fact that for a group R to be feasible, all subgroups $R' \subset R$ must also be feasible. It first computes feasible groups of size one, and then it iteratively creates larger groups by combining the feasible groups from the previous iteration with the feasible groups of size 1. The critical parts of the function are the calls to the f function. Note that to compute an optimal system plan, we need to implement the f procedure to not just check the feasibility but also to return an optimal plan for the given group, although it is not explicitly shown in the pseudocode of Algorithm 3. However, this has no impact on the worst-case complexity, as in the worst case, there will be no or only one feasible sequence of pickup and drop-off actions for each group.

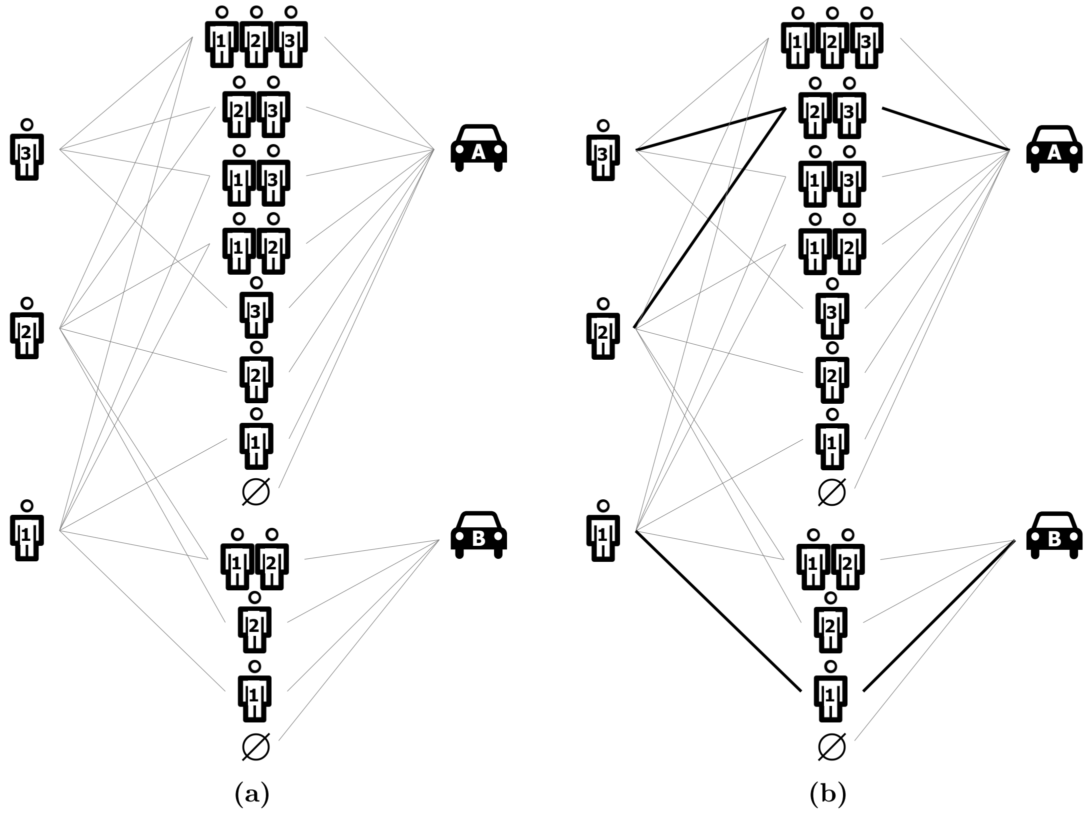


Figure 2.1: Example of the VGA method solving DARP with three passengers and two vehicles. In Figure 2.1a, we show all groups of requests for each vehicle. The lines between the request (left) and the group (middle) denote the membership in the group. The lines between the groups and vehicles denote feasible group assignments. In Figure 2.1b, the final assignment between vehicles and groups is shown (bold lines).

Algorithm 3 Function `generate_groups` that generates groups for vehicle a. The boolean-valued function $f(R, a)$ evaluates to true if a feasible plan serving all requests in group R exists for vehicle a.

Input: A vehicle a and the set of requests D.

```
for $r \in D$ do                                                    ▷ generate groups of size 1
    if $f(\{r\}, a)$ then
        $\gamma_a^1 \leftarrow \{\{r\}\} \cup \gamma_a^1$
$k \leftarrow 1$
while $\gamma_a^k \neq \emptyset$ do                                ▷ generate remaining groups
    $\gamma_a^{k+1} \leftarrow \emptyset$
    checked $\leftarrow \emptyset$                                   ▷ not check groups repeatedly
    for $R \in \gamma_a^k, \{r\} \in \gamma_a^1$ do
        if $(R \cup \{r\}) \notin$ checked and $\forall R' \subset (R \cup \{r\}), |R'| = k : R' \in \gamma_a^k$ then
            if $f(R \cup \{r\}, a)$ then
                $\gamma_a^{k+1} \leftarrow (R \cup \{r\}) \cup \gamma_a^{k+1}$
        checked $\leftarrow$ checked $\cup (R \cup \{r\})$
    $k \leftarrow k + 1$
$\gamma_a \leftarrow \{\emptyset\} \cup \gamma_a^1 \cup \gamma_a^2 \cup \cdots \cup \gamma_a^k$
```

At the end of the group generation step, we have a set of feasible groups for each vehicle, as it is illustrated in Figure 2.1a.

Vehicle-group Assignment

In the second part of the VGA method, we need to assign the vehicles to groups such that: a) each vehicle is assigned to no more than one group, b) each request is a part of exactly one group, and c) the total traveled distance is minimized. We formulate this problem as an integer linear program (ILP) with binary variables x for each feasible group resulting from the group generation procedure. If $x_R = 1$, then all requests in the group R are served by the vehicle to which this group belongs following the associated plan. The problem is formulated as follows:

Problem 2.2 (Vehicle-group Assignment).

$$\min \sum_{R \in \gamma} f_{\text{tc}}({p_R}^*) x_R,$$

subject to

$$\sum_{R\in\gamma_i} x_R = 1 \quad \forall i = 1, \cdots, |V| \tag{2.1}$$

$$\sum_{R\in\gamma} \mathbf{1}_R(r) x_R = 1 \quad \forall r \in D. \tag{2.2}$$

Constraint 2.1 ensures that only one group can be assigned to each vehicle. Constraint 2.2 guarantees that each request is served by exactly one vehicle plan. The indicator function $\mathbf{1}_R(r)$ is equal to 1 if the request r is a member of the group R and 0 otherwise.

By solving the above-described ILP, we obtain an optimal assignment of vehicles to feasible groups (see Figure 2.1b for example assignment).

Complexity

The exact complexity of the VGA method is the sum of the complexities of its two subproblems: group generation and vehicle-group assignment. We do not try to analyze the complexity of the vehicle-group assignment part of the algorithm, as the algorithmic complexity of solving the ILP can vary depending on the ILP solver used. However, as the ILP problem is an NP-hard problem (again, a generalization of one of the 21 NP-complete problems (Karp, 1972)), we know that the complexity of vehicle-group assignment is superpolynomial (supposing P $\neq$ NP). Regarding the ILP formulation of the vehicle-group assignment, we can at least remark here on the number of binary variables and the number of constraints. The number of binary variables is equal to the number of feasible groups generated by the group generation procedure: $\sum_{i=1}^{|V|} |\gamma_i|$. The number of vehicle constraints (Constraint 2.1) is equal to the number of vehicles, and the number of request constraints (Constraint 2.2) is equal to the number of requests.

We will now focus on the complexity of the group generation procedure. The group generation complexity is dominated by the need to verify the feasibility of a group, represented by the call of function $f(R, v)$. Solving this function, in fact, equals solving a single-vehicle DARP, which is an NP-hard problem (see Section 2.2.3). For each group, we need to, in the worst case, check the feasibility of all permutations of actions for which it holds that each drop-off action is after the corresponding pickup action. The total number of permutations of actions is $(2|R|)!$. However, the overall complexity is much lower, as most of the permutations have at least one drop-off action before the corresponding pickup action. To quantify this, we need to realize two facts: 1) for each request half of the permutations have the drop-off action before the pickup action, and 2) the misordering of actions is independent between requests. Knowing this, we can now compute the number of valid permutations as the number of all permutations divided by half by each request:

$$\frac{(2|R|)!}{2^{|R|}} \tag{2.11}$$

The feasibility function is called for each group. The number of possible groups of size n is $\binom{|D|}{n}$ (all combinations of requests of size n). In the worst case, the maximum group size is $|D|$, so the complexity of the group generation procedure is:

$$O\left(\sum_{n=1}^{|D|}\left(\binom{|D|}{n}\frac{(2n)!}{2^n}\right)\right) \tag{2.12}$$

Finally, we will discuss the practical complexity of the VGA method. The DARP problem instances appearing in the context of large-scale MoD systems tend to have structural properties that are beneficial to the VGA algorithm. Specifically, since large-scale MoD systems are designed to provide quality of service comparable to using a private vehicle, the maximum waiting at pick-up is usually constrained to be less than a few minutes, and similarly limited is the maximum delay at destination. Such tight pick-up and drop-off time window constraints are used by the VGA algorithm to prune the feasible solution space. Moreover, the instances typically have a short time horizon. Therefore, in practice, the maximum group size that requires a feasibility check tends to be relatively small. Even more importantly, the maximum feasible group size is independent of the number of requests as we increase the number of requests by enlarging the operational area and not by creating unrealistically dense demand in a small area. As a result, we can say that in practical large-scale DARP instances of short time horizon, there is a constant K such that the maximum feasible group size is less than K, and the complexity is then:

$$O\left(\sum_{n=1}^{K}\left(\binom{|D|}{n}\frac{(2n)!}{2^n}\right)\right) \tag{2.13}$$

Additionally, limited group size means the total number of feasible groups that result from the group generation and need to be processed by the vehicle-group assignment tends to be within the grasp of existing ILP solvers. Under such conditions, the VGA method can generate optimal results in practical computation time, which is demonstrated by the VGA application results presented in this thesis (in Chapter 4, but also in Chapters 3 and 5).

2.4 Classical and MoD DARP

Originally, the DARP was formulated to solve vehicle dispatching in the context of Dial-a-ride (DAR) systems, which are on-demand systems to solve elderly people or people with disabilities (Guenther & Givens, 1970; National Research Council (U S.), Highway Research Board, 1971). This is still the dominant application of the DARP in the operational research community (Ho et al., 2018). So how can we claim to solve DARP in the context of MoD systems? As we mentioned before, the DAR systems are principally the same as MoD systems. Both systems provide on-demand transportation services and use the same basic constraints and objectives. There can be small differences in the formulation of the service quality constraints (e.g., time windows vs maximum delay), but these are not fundamentally bound to the real problem, they are rather research community conventions.

However, apart from minor formulation differences, which were discussed in Section 2.2.3, there are big differences in the instance characteristics. First, in MoD systems, instances usually have only short time horizons τ, especially the more typical online instances. While the classical DARP instances usually cover a whole day (Cordeau, 2006; Cordeau & Laporte, 2003; Ropke & Pisinger, 2006), in MoD, the instances are usually solved online and thus the time horizon is limited to minutes or even seconds (Alonso-Mora et al., 2017; Fiedler et al., 2018; Jung et al., 2015). Second, the demand in DAR systems is usually much smaller than in MoD systems. In New York City, for example, there can be as many as 100 000 (NYC Taxi & Limousine Commission, 2018) active taxis during peak traffic hours, at least *three orders of magnitude* more than in the largest traditional DARP instances (Cordeau, 2006; Cordeau & Laporte, 2003; Ropke & Pisinger, 2006). Finally, the time window length ω is usually much shorter in MoD systems than in classical DARP instances. In MoD, passengers' time constraints are much more strict because MoD systems are usually focused on the general population who assumes immediate service. Compared to that, a typical DAR system user is expected to be more flexible.

We discuss these differences here because they have far-reaching consequences for the complexity of the DARP. It is clear that the larger demand scale in MoD systems leads to a higher combinatorial complexity compared to the classical DARP. However, it is not so intuitive that the remaining two differences also impact complexity significantly, this time, in the opposite direction. The long time windows create more flexibility: with longer time windows, more requests can be served by the same vehicle, and higher occupancy increases the complexity dramatically. The long time horizon, on the other hand, increases the number of requests that can be served sequentially by the same vehicle. Moreover, while each request can be transported simultaneously only with a limited number of other requests that are geographically close, sequentially, the number of requests that can be served by the same vehicle is virtually unlimited. Therefore, a long time horizon has also drastic consequences for the complexity of the DARP.

To summarize, the large scale of the demand in MoD systems increases the complexity compared to the classical DARP, but the shorter time horizon and time windows act in the opposite direction. Although we do not provide any mathematical analysis of the impact of these factors, they can be observed in the sensitivity analysis of the MoD simulation studies presented in Chapter 4. These results show that even large instances can be solved optimally and thus, the MoD DARP instances are less complex than the classical DARP instances, despite being many orders of magnitude larger.

CHAPTER 3

Instances for MoD-related problems

Most problems or problem areas require some benchmark instances to evaluate new methods, perform case studies, or experiment with new ideas. Mobility-on-demand (MoD) systems are no exception: many MoD-related problems like dispatching or fleet-sizing require problem instances in the same form that is needed for DARP (specified in Section 2.2.2). In this chapter, we show that currently, there are no established publically available instances suitable for modeling large-scale MoD systems. Moreover, we describe the challenges related to the creation of such instances which can explain the lack of them. Finally, we propose a methodology for creating large-scale MoD instances based on real travel demand data, present the instances created using this methodology, and evaluate them using two known established methods for solving the dial-a-ride problem (DARP).

3.1 Existing Public Instances

Before we start with the description of the proposed instances, we first describe the currently used instances. In the next section, we describe the *classical instances*, which are small-scale randomly generated instances containing points in the Euclidean plane. These instances are typically used in the operational research literature. Next, we describe the *modern instances* used in the transportation literature (including the MoD-related research), which are much larger and typically based on real travel demand data[1]. We will discuss the problems and

[1]The classical and modern specifiers here are not established terms and we introduce them purely to easily distinguish between those two instance types/groups. Calling the instances used in MoD transportation as MoD instances could be confusing here, as the classical DAR systems also belong to the category of on-demand systems. However, later in the paper we consider only modern instances and refer to them as MoD instances or just instances

This chapter is based on the following conference paper:

(Fiedler & Mrkos, 2023b) Fiedler, D., & Mrkos, J. (2023b). Large-scale Ridesharing DARP Instances Based on Real Travel Demand, 2750–2757. https://doi.org/10.1109/ITSC57777.2023.10422146

For all author's publications together with the citation counts and research contributions, see Appendix A.

Publication	Instances	Requests	Vehicles	ω [min]	τ [min]
Cordeau and Laporte (2003)[a]	20	24–144	3–13	15-90	1440
Cordeau (2006) & Ropke et al. (2007)[b]	42	16–96	2–8	15	480-720

[a] https://chairelogistique.hec.ca/data/darp/tabu/
[b] https://chairelogistique.hec.ca/data/darp/branch-and-cut/

Table 3.1: Classical DARP instances. Here ω is the time window length and τ is the time horizon (see Section 2.2.2 for the definitions). Other instances for special DARP variants are usually based on the second set of instances. Examples of such specialized instances can be heterogenous DARP instances by Parragh (2011) or multi-depot heterogenous DARP instances by Braekers et al. (2014).

challenges related to these instances. Finally, we describe an alternative approach to creating realistic large-scale DARP instances from generated travel demand data.

3.1.1 Classical DARP Instances

The emergence of research related to the DARP in the 1970s brought the first DARP instances (Psaraftis, 1980). These hand-made instances were located on the Euclidean plane and contained less than 20 requests.

At the beginning of the 21st century, a handful of new DARP instance datasets were published (see Table 3.1). Each of those datasets contains tens of instances and the instances are an order of magnitude larger than the instances by Psaraftis (1980), containing tens of requests. However, the main properties of the instances stayed the same: the instances are located on the Euclidean plane, so there is no road network or travel time model considered, all vehicles start from a single depot, and the requests are generated randomly without considering any real travel demand data. Because of this and the popularity of these datasets, we call them *classical* DARP instances. We can see an example instance from the dataset by Cordeau and Laporte (2003) in Figure 3.1.

In the last twenty years, more sophisticated large-scale instances based on real demand were developed to provide better insights into real-world MoD problems. These instances are described in the following section. Nevertheless, a large amount of the articles from the last two decades still use the classical instances (Cordeau, 2006; Cordeau & Laporte, 2003; Gschwind & Drexl, 2019; Ham, 2021; Kirchler & Wolfler Calvo, 2013; Masmoudi et al., 2020; Parragh, 2011; Ropke & Cordeau, 2009; Ropke & Pisinger, 2006). However, these articles study the original MoD systems for the transportation of people with disabilities (DAR systems), and therefore, at least the instance size is realistic.

3.1.2 Modern Instances for Modeling Mobility Problems

With the emergence of large-scale mobility-on-demand (MoD) systems operated by transportation network companies (TNCc), the need for new instances arose because the charac-

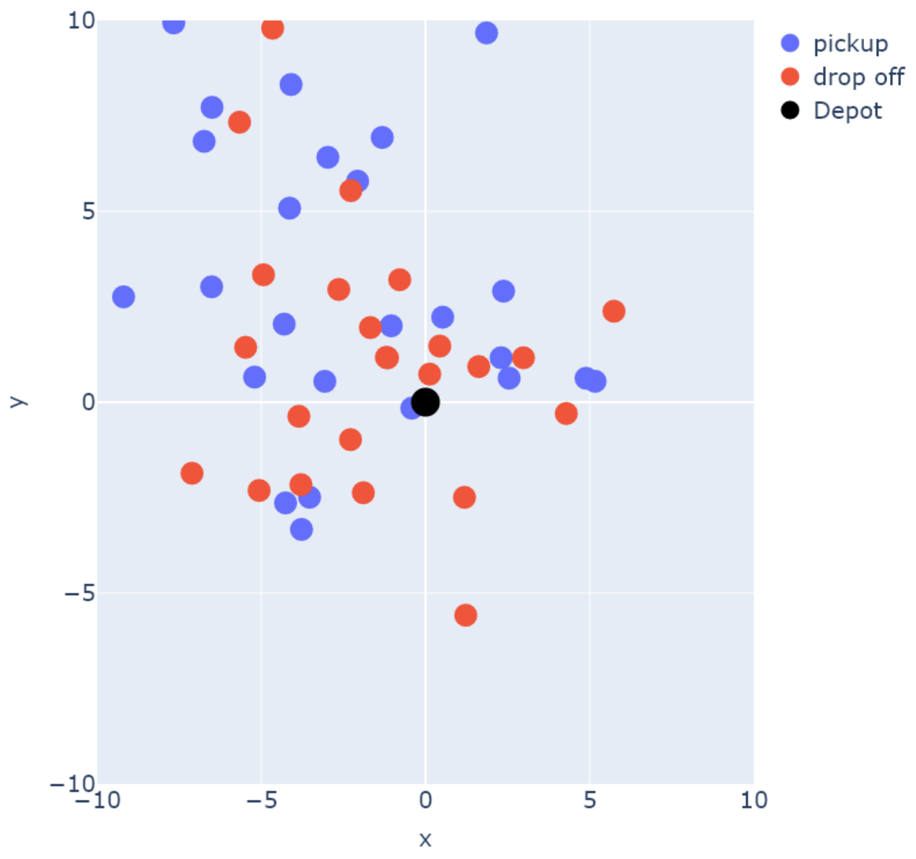


Figure 3.1: Example of a classical DARP instance from the dataset by Cordeau and Laporte (2003). The instance contains 24 requests and 3 vehicles.

teristics of MoD DARP are different from the classical DARP. In fact, all main properties (scale, τ, and ω) differ from the classical DARP. In this section, we discuss these differences, the current state of MoD problem instances, and the problems and challenges related to the creation of such instances. Note that while modern instances are rarely referred to as DARP instances, they still have the same structure as the DARP instances as defined in Section 2.2.2.

Difference between Classical and Modern Instances

The first notable difference between the instances representing MoDs operated by TNCs and the classical instances is size. In New York City, for example, there can be as many as 100 000 active taxis during peak traffic hours, at least three orders of magnitude more than in the largest classical instances (NYC Taxi & Limousine Commission, 2018).

Because of the large scale and also to be in line with the real operation, the vehicles do not start in a single depot in modern instances. Vehicles are instead distributed throughout the area to serve the demand with the shortest possible delay while supporting large areas. At the same time, having a fleet of thousands of vehicles concentrated in one depot like in classical instances would be unrealistic. Therefore, in modern instances, there are usually many vehicle depots or the vehicles are free-floating, i.e., each vehicle has its own initial position.

Modern instances also differ from the classical ones in two key properties. First, the instance horizon τ is usually shorter. For the transportation of elderly or disabled people, which is the original application of DARP, the time horizon is usually 24 hours. In other words, it was assumed that the users of the system order the transportation no later than the day before the trip. In contrast, the TNC-operated MoD systems are usually used for immediate transportation, so the time horizon is usually much shorter. In fact, most of the MoD systems work in an online mode, where the request's desired pickup time equals the request's announcement time. Even instances with a very short time horizon are sufficient to experiment with these online DARP problems.

Second, the time windows ω are much shorter in modern instances. The reasons for that are the same as in the case of the time horizon: these systems are used for immediate transportation, so the users do not have the flexibility. At the same time, unlike the classical DAR systems operated by municipalities, the current MoD systems are operated by private companies and thus are much more expensive for the users. Therefore, there are higher requirements regarding the quality of service, resulting in shorter time windows.

Finally, due to these different characteristics of modern MoD systems, we do not need some parameters/constraints present in classical instances. The limit on the maximum duration of vehicle plans is not necessary, as the time horizon of the instance is short. Similarly, we do not need the maximum ride time of passengers because the time windows are short.

Emergence of Instances Based on Real Data

Apart from the large-scale MoD systems operated by TNCs, the digitalization of transportation systems has also brought large-scale datasets of travel demand and other transportation-related data. Because of this, many research works related to MoD systems now use large-scale instances based on real travel demand datasets (Alonso-Mora et al., 2017; Beirigo et al., 2022; Beojone & Geroliminis, 2021; Bischoff et al., 2017; Burns et al., 2013; Choi et al., 2023; Fiedler, Čertický, Alonso-Mora, et al., 2022; Guo et al., 2021; Hyland & Mahmassani, 2018; Kondor et al., 2022; Kumar et al., 2023; Lokhandwala & Cai, 2018; Los et al., 2022; S. Ma et al., 2015; Santi et al., 2014; TU et al., 2019; Tuncel et al., 2023; Vazifeh et al., 2018; Wallar et al., 2019b; Wollenstein-Betech et al., 2021; B. Yu et al., 2017; Zhan et al., 2022).

Creating MoD instances using real demand data offers several advantages. Firstly, it provides a realistic representation of both the scale and distribution of the demand. Secondly, incorporating the temporal changes in demand into the problem instances allows for a more accurate reflection of dynamic patterns and variations in service requirements over time. Lastly, by utilizing road network travel times instead of relying solely on Euclidean distance, the computed travel cost aligns more closely with actual transportation conditions, enhancing the realism of the instances.

Nevertheless, even with the availability of large-scale demand datasets, there are many problems and challenges related to the creation of modern instances. These problems and challenges are described in the following section.

Problems and Challenges

Apart from apparent advantages, the large realistic problem instances present new challenges. Instead of a simple random instance generation which we know from the classical instances, we must process and transform the travel demand datasets into the instance data. Moreover, in recent years, data providers started to obfuscate the published travel demand datasets to protect the privacy of users and drivers. This means that the request origin and destination locations and, sometimes, even request times are not specified precisely. Instead, locations are binned into zones and times into time intervals. Thus, we must first generate trip locations and times to create instances from obfuscated data. The absence of a standardized format for open data across various cities further complicates the situation; a procedure for creating instances from one dataset cannot be applied to other datasets without significant changes. Finally, we need other datasets for every location where we wish to create a problem instance. At the very least, researchers must process the road network data and, possibly, some dataset of travel speeds to obtain realistic travel times.

To avoid some of these pitfalls, many works use the NYC travel demand datasets published before the introduction of privacy protection rules in 2016[2]. However, the Manhattan dataset has a highly distinctive geography and unusually high demand density. Moreover, due to the 2016 cutoff, new phenomena, such as the displacement of traditional taxi services by transportation network companies or the COVID pandemic, are not captured in datasets

[2]In (Fiedler & Mrkos, 2023a), we claimed that the change took place in 2014. We made this mistake most likely because the 2014 dataset is the most used one in the transportation studies. Note that the exact date is hard to determine as the NYC Taxi and Limousine Commission replaced the previously published data with the obfuscated ones even for the years before the new privacy policy. However, recently, we have found the original `csv` datasets with both precise origins/destinations (March) and with origins/destinations specified by zones (July) for 2016, so we believe that this is the year when the policy has changed.

based on the pre-2016 Manhattan demand data. This is important since many new methods are evaluated *only* using the NYC/Manhattan demand dataset (Alonso-Mora et al., 2017; Beirigo et al., 2022; Fiedler, Čertický, Alonso-Mora, et al., 2022; Haliem et al., 2021; Santi et al., 2014; Seo & Asakura, 2022; Thangaraj et al., 2017; Vazifeh et al., 2018; Wallar et al., 2019b; W. Zhang et al., 2023). As a result, the field of DARP research runs the risk of overfitting specific problem instances that lack generalizability to different locations or current travel patterns. This can result in misleading conclusions and unrealistic expectations when evaluating DARP solution methods.

The last issue with modern instances is standardization. At first glance, it may appear that using old Manhattan instances at least provides the standardization we know from the classical instances (Cordeau, 2006; Cordeau & Laporte, 2003; Ropke et al., 2007). The ability to compare methods to previously established ones without the need for re-implementation is a crucial factor for productivity. However, there are no publicly available instances for Manhattan or other cities. Instead, every work presents its own instances based on the demand datasets (private, or public) without publishing them; thus, their results are hardly comparable or replicable.

3.1.3 Generated Travel Demand

The last option for creating MoD instances is to use generated travel demand. Unlike classical instances where the travel demand is generated uniformly, researchers also investigate the generation of realistic travel demand. This is useful for performing realistic case studies if we cannot use real travel demand data. The historical travel demand data are available only for a limited number of cities, therefore, it is likely that a researcher will not be able to obtain the data for the city or area of interest. Moreover, even if the researchers have access to the data, they may not be able to use it in their publications due to privacy concerns(ATOCKAR, 2014; Culnane et al., 2019). It was repeatedly demonstrated that it is possible to infer real travel diaries even when some privacy-preserving measures like anonymization or obfuscation are applied (Fiore et al., 2020; Gambs et al., 2014; Srivatsa & Hicks, 2012). For these two reasons, it makes sense to develop methods for generating realistic travel demand, even in the era of big data.

Traditionally, demand models were centered around individual trips which were generated in stages. The dominant approach was the four-step model whose stages are a) *trip generation*: the generation of incoming and outgoing demand for each zone, b) *trip distribution*: the generation of the number of trips between each pair of zones, c) *mode choice*: the assignment of the mode of transport, and d) *route assignment*: the decision of the route (Hensher, 2007).

Nowadays, the trip-centric approach is considered obsolete (Ben-Akivai et al., 1996; Hensher, 2007) and most of the literature is focused on users and their daily activities or travel itineraries (Kapp et al., 2023). *Activity-based models* (Hensher, 2007) play a prominent role in this family of models. Activity-based models employ so-called activities (e.g., work, shop, sleep) and their sequences to represent the transport-related behavior of the population (Čertický et al., 2015). Travel demand then occurs due to the agents' necessity to satisfy their needs through activities performed at different places at different times.

These activities are arranged in time and space into sequential daily schedules. Trip origins, destinations, and times are endogenous outcomes of activity scheduling. The activity-based approach considers individual trips in context and therefore allows representing realistic trip chains. In recent years, many activity-based models were studied (Auld & Mohammadian, 2012; Auld et al., 2016; Bassolas et al., 2019; Bösch et al., 2016; Čertický et al., 2015; Drchal et al., 2016; Lu et al., 2015; Neumann et al., 2012), see for example the survey by Tajaddini et al. (2021) for more details.

Finally, there are now great opportunities to use big data from transportation network companies, mobile phone operators, navigation applications, or social networks. With the recent advances in machine learning (ML), it is possible to replace some expert-defined parameters or parameters calibrated by small survey datasets and improve the realism of the generated travel demand. Many recent studies propose to use of ML instead of expert-designed components of activity-based models (Drchal et al., 2019; Widhalm et al., 2015; Yin et al., 2018). Other studies also propose to use of ML to create user-centric demand models but they do not use the activity-based approach(Badu-Marfo et al., 2022; Berke et al., 2022; Ouyang et al., 2018; Pang et al., 2017; Toole et al., 2015). For a comprehensive overview of these generative demand models, see the survey by Kapp et al. (2023)

To conclude this section we look at the use of generated travel demand in transportation studies. Compared to the massive amount of research that uses historical travel demand data (see Section 3.1.2), the use of generated travel demand is less common, and to our best knowledge, limited to the activity-based models (Balac et al., 2019; Bischoff & Maciejewski, 2016; Hörl et al., 2019; Maciejewski & Bischoff, 2018; Marczuk et al., 2015; Najmi et al., 2017; X. Yu et al., 2022)[3]. Pure ML models, while trending in ML research, are yet to be adopted by the transportation community. Moreover, the generated instances, similarly to the instances based on historical demand, lack the standardization necessary for the comparison of methods and reproducibility of results.

3.2 New Large-scale DARP Instances Based on Real Demand

In this section, we present new large-scale DARP instances based on real travel demand data from three cities: New York, Chicago, and Washington, DC. The instances are based on recently collected data, capturing the latest developments in the MoD field. Moreover, we present a detailed description of the process of creating these instances, supported by the source code needed to create them. Finally, we evaluated the instances using two known methods for solving DARP to demonstrate the advantages of using multiple areas and instance configurations. The methods used are 1) a simple construction heuristic: the insertion heuristic (Campbell & Savelsbergh, 2004), and 2) an optimal solution method: the vehicle-group assignment method (Alonso-Mora et al., 2017). Together with the proposed instances, we distribute full solutions computed by these methods in the instance repository [4].

[3]Simple uniform models or arbitrary toy examples are not considered here

[4]https://github.com/aicenter/Ridesharing_DARP_instances

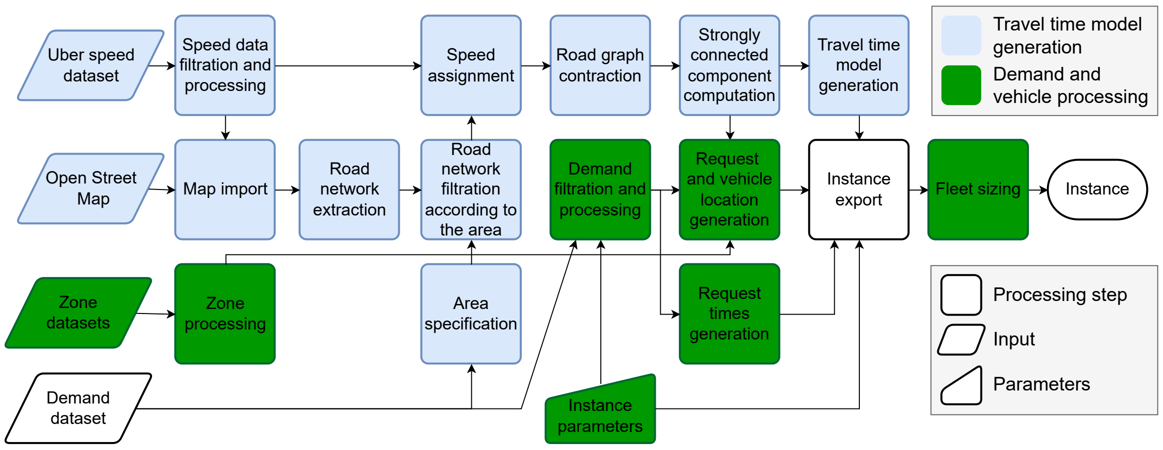

Uber speed dataset
Speed data filtration and processing
Speed assignment
Road graph contraction
Strongly connected component computation
Travel time model generation
Travel time model generation
Demand and vehicle processing
Open Street Map
Map import
Road network extraction
Road network filtration according to the area
Demand filtration and processing
Request and vehicle location generation
Instance export
Fleet sizing
Instance
Zone datasets
Zone processing
Area specification
Request times generation
Processing step
Input
Parameters
Demand dataset
Instance parameters

Figure 3.2: Flow chart of the instance creation process.

3.2.1 Methodology

The instances we present are large ridesharing DARP instances based on real travel demand. Each instance contains a) demand in the form of a list of requests for transportation between two points in the road network, b) a set of vehicles represented by their starting locations, and c) a model of travel time between any two locations in the road network.

We present the travel time computation for the travel time model f_t in Section 3.2.1. We use open travel data specific to each city to generate the demand D and vehicles V. This is described in Section 3.2.1. Finally, in Section 3.1.2, we explain the differences between classical DARP instances and the proposed instances for ridesharing DARP. The flow chart covering the whole process of creating an instance is in Figure 3.2.

Computing Travel Time

In classical DARP instances, the request origin and destination locations, as well as the start locations for vehicles, are coordinates in Euclidean space. The travel time between any two points is then the distance between these points. However, real travel times are much more complex: they are not symmetric and cannot be calculated purely from origin-destination coordinates. To approximate the real travel time, we use a road network with assigned speeds for each road segment and compute the shortest path between locations in the road network. Note that we cannot use the demand data for a more precise approximation of travel times (like, for example, Santi et al. (Santi et al., 2014)) because the recent demand datasets are obfuscated, and thus, we cannot match the trip origin/destination points to the nearest intersection.

We base the travel time model on two datasets: the OpenStreetMap[5] for the road network, and the UberMovement dataset[6] for speeds. The shortest-path computations are expensive, which is exacerbated by the fact that most methods for solving DARP need to

[5]https://www.openstreetmap.org/

[6]https://movement.uber.com/. Unfortunately, this open dataset is no longer available.

compute travel times frequently. Therefore, the travel times are usually precomputed in a distance matrix to reduce the overhead. However, to keep the size of the matrix manageable, we need to discretize the road network to limit the number of origin-destination pairs. In this work, we use intersections as locations.

Note that in Section 2.2.1, we concluded that the travel time function can accept directly the pickup and drop-off actions, which would result in a much smaller distance matrix. This is definitely possible in this case as well. The reasons why we modeled travel times for the whole road network are purely technical: we used an existing program for the distance matrix computation that accepts the whole road network as an input. Nevertheless, modeling travel times for the whole road network has also an advantage: it allows us to use the same travel time model for all instances located in the same area, regardless of the demand locations.

The travel time model for each area was created through the steps shown in light blue in Figure 3.2. The crucial parts of the process are:

1. **Speed data preparation** for the area, date, and time of interest.
2. **Area specification**: the instance area is specified as a convex shape such that all the demand in the instances we plan to generate lies inside this shape,
3. **Import of the selected map** from the OpenStreetMap dataset according to the map state at the time when the speed dataset was created,
4. **Filtration of the map** so that it contains only road network,
5. **Speed assignment** to roads according to the speed dataset,
6. **Road graph contraction** by elimination of all nodes that are not intersections.
7. **Largest strongly-connected component** selection to remove unreachable "islands" either real or artifacts of map filtration.

After the largest strongly connected component computation, the road network processing is finished, and we can produce the travel time model $f_{\mathrm{tt}}(a,\ b)$.

Demand and Vehicles Processing

This section provides a brief description of the process for transforming a travel demand dataset into demand D and vehicle data V for the DARP instance.

The initial step in the creation of the demand component D involves selecting the date, start time, and end time. These parameters are then used to extract the relevant records from the travel demand dataset for a specific city.

As previously mentioned, the travel demand data are now obfuscated by all data providers due to privacy concerns. Consequently, to generate the instance, we must devise a method for generating the demand locations and, in some cases, the demand origin time. In both cases, we sample a uniform distribution across nodes in the processed road network (Section 3.2.1) within the designated zone specified in the dataset and across the dataset's pickup time interval, respectively. As neither the zones nor the time intervals are large (see Table 3.2), the uniform distribution should not cause an excessive error.

Finally, we also need to generate vehicles and determine their starting positions. To accomplish this, we sample from the demand dataset drop-off points. This approach ensures that we obtain realistic vehicle positions, while not exploiting the knowledge of the demand. The vehicle fleet size in each instance is chosen as the lowest number of randomly selected vehicles that can service all requests when solving by insertion heuristic. This number is then increased by 5 % to introduce a buffer for cases where the insertion heuristic would find the unique optimal solution.

3.2.2 DARP Instances

We generated instances for four areas: New York City, Manhattan, Chicago, and Washington, DC. The data sources for those areas are listed in Table 3.2. Each of the generated

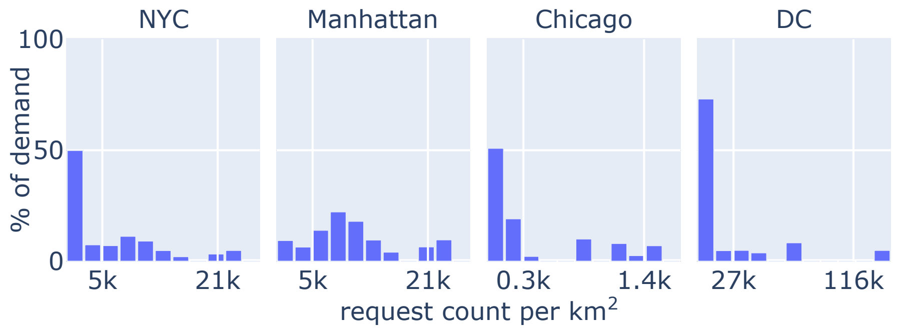
Table 3.2: Demand and zone data sources[a]

Area	**Demand data source**	**Zone data source**	**Req. times**	**Req. per h per km^2**	**Avg. zone area [km^2]**
NYC	NYC Taxi and Limousine Commission	NYC taxi zones	exact	27.52	3.01
Manhattan				475.21	1.04
Chicago	City of Chicago	Census tracts and community areas	gen.	1.17	1.34
DC	City of Washington, DC	Master Address Repository	gen.	3.65	0.01

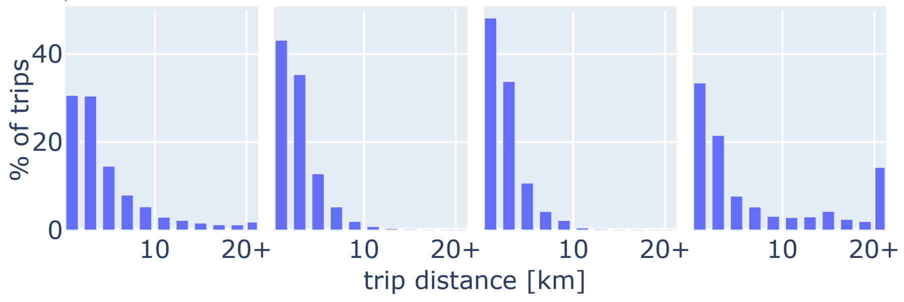
[a] The download links for all data sources can be found in the instance repository: https://github.com/aicenter/Ridesharing_DARP_instances

areas has different characteristics of the travel demand. In Figure 3.3, we can see a comparison of the areas using two quantities: demand density and trip length. Note that the characteristics differ significantly between the areas. A typical travel request in Manhattan originates in a zone with about 10 000 requests per km per day. In Chicago, a typical request originates in a zone with a demand density of two orders of magnitude lower. A similar difference can be seen in the average trip length, which is low in Manhattan and Washington, DC, and high in Chicago.

For each area, we determined the boundary differently. For Manhattan and Washington, DC, we used the administrative boundary. However, the administrative boundaries of the remaining areas do not match the demand zones, so we generated the boundary as convex hulls around the demand zones. For Chicago, we only considered zones with at least 30 requests to reduce the area and increase the density of the demand. The area boundaries and zones for all areas are presented in Figure 3.4. Note that the DC zones have irregular shapes. This is because the demand zones in DC are specified as Master Address Repository

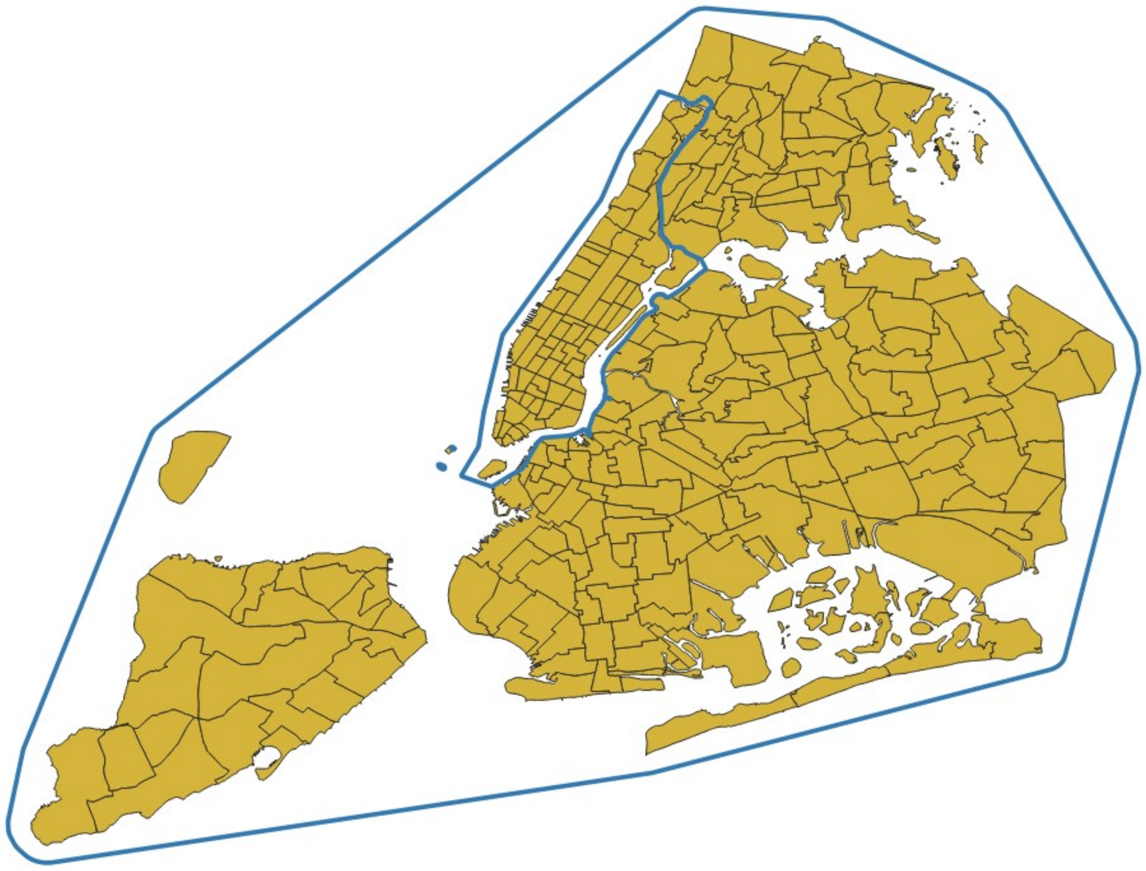


(a) Demand density histogram. The x-axis represents the density of the daily demand in the request's origin zone (Note that it has a different scale for each area).

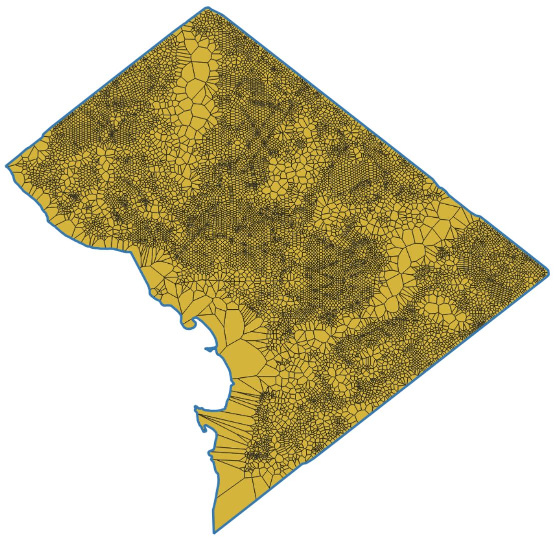


(b) Trip Length Histogram

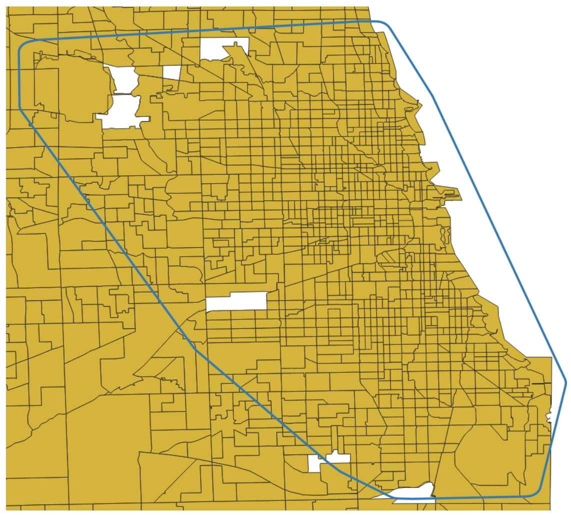
Figure 3.3: Statistics of the daily demand in each instance area.



(a) New York City and Manhattan

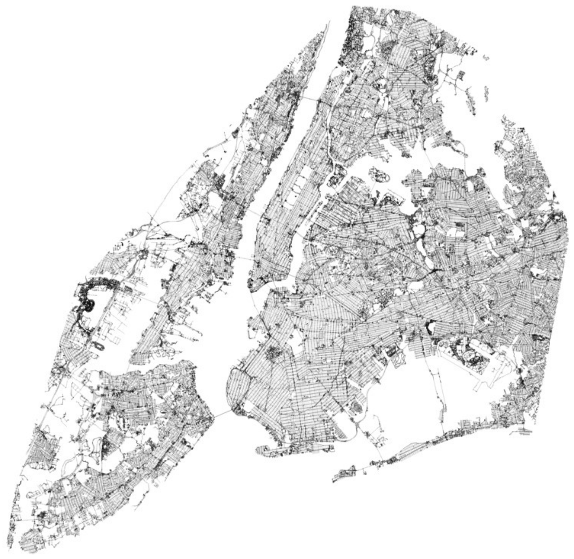


(b) Washington, DC



(c) Chicago

Figure 3.4: Area boundaries (blue) and zones (gold). Only the zones within the area boundary are used to generate the instances.


(a) New York City

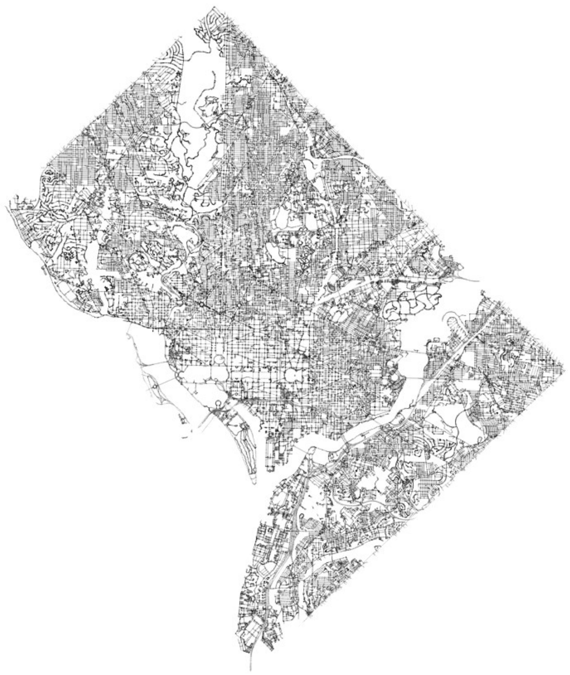

(b) DC

Figure 3.5: Example road networks

Table 3.3: Road network statistics.

Area	**Node Count**	**Edge Count**	**Road length [km]**	**Area [km^2]**
NYC	113411	281278	27721	1508
Manhattan	6382	13455	1329	87
Chicago	152653	413830	31982	1004
DC	33230	84788	5877	181

zones[7]. As we do not have access to boundaries for these zones, only their centroids, we generated the zone boundaries as Voronoi cells based on these centroids.

The road networks were processed according to the steps described in the methodology section. Only the roads within the area boundary (see Figure 3.4) were used to generate the road network for each area. Two example road networks are visualized in Figure 3.5. The statistics for these road networks are summarized in Table 3.3.

Each edge in the road network of all instances has speed associated with it. The speed data were sourced from the Uber Movement[8] dataset for the New York City and Manhattan areas. This data is visualized in Figure 3.6. We lacked a data source with a similar level of detail for Chicago and Washington, DC, so we associated an average speed from the Uber

[7]Method of address standardization used in DC, https://octo.dc.gov/node/715602
[8]https://movement.uber.com/

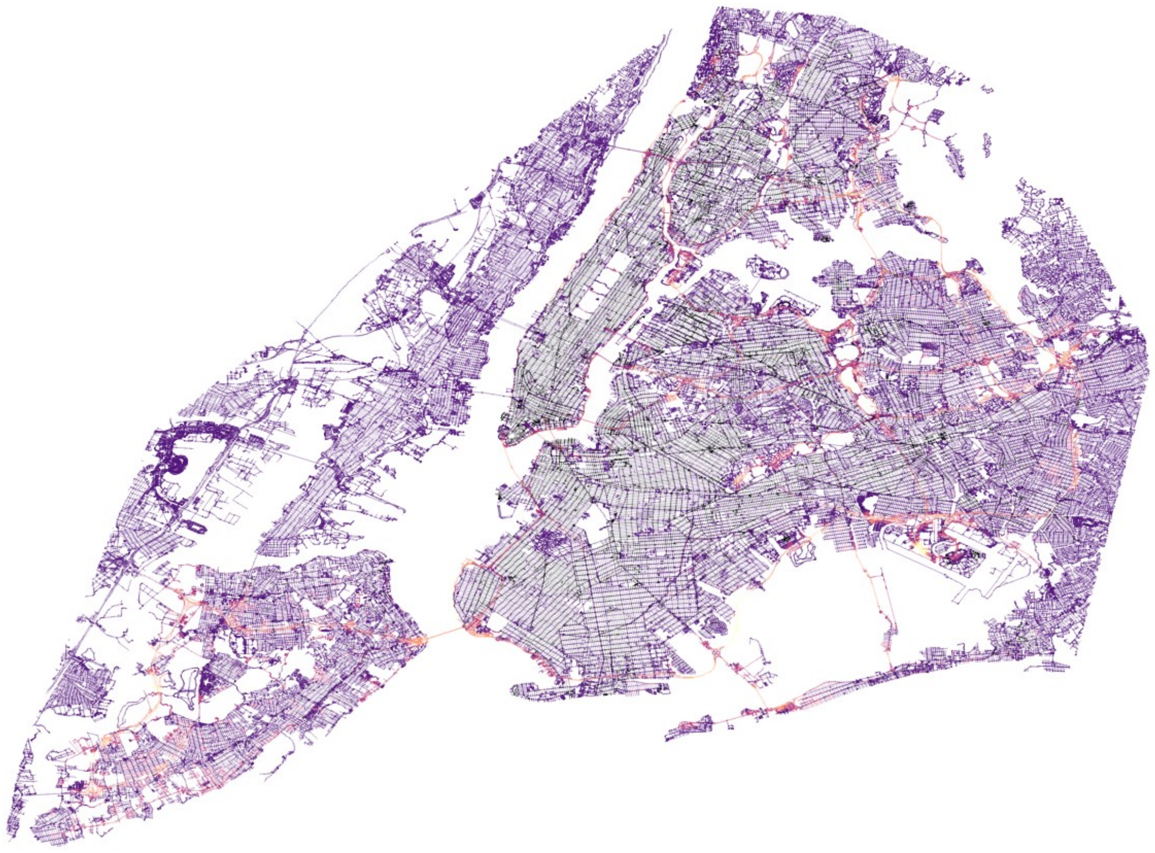


Figure 3.6: Speed map of New York City. The lighter color translates into higher speeds.

Movement dataset with all edges.

Finally, by combining the road graph with the speed data and shortest path planner, we generated a distance matrix for each area that determines the travel time between any two locations (intersections in the road graph) and comprises the travel time model $f_{\mathrm{tt}}(a,\ b)$. This matrix is enough to very quickly provide any necessary travel time information for DARP solvers and similar algorithms; the map itself is an intermediate product provided as supplemental material for visualization purposes and the user's convenience.

The dataset we present currently contains 96 instances, 24 for each area. The parameters according to which these instances were generated are listed in Table 3.4. Note that instances with different parameters can be generated by following our methodology; this is just an example set with parameters set to values relevant (to our best knowledge) to MoD.

In addition to the travel time model in the form of distance matrix $f_{\mathrm{tt}}(a,\ b)$, each instance contains a list of travel requests D, a list of vehicles V, and its configuration. An example of the proposed instance is in Figure 3.7. The instances are complemented by additional meta-data, such as the timestamp of the start time of the demand, location name, road graph, travel speed data, and other supporting information useful for visualization and analysis.

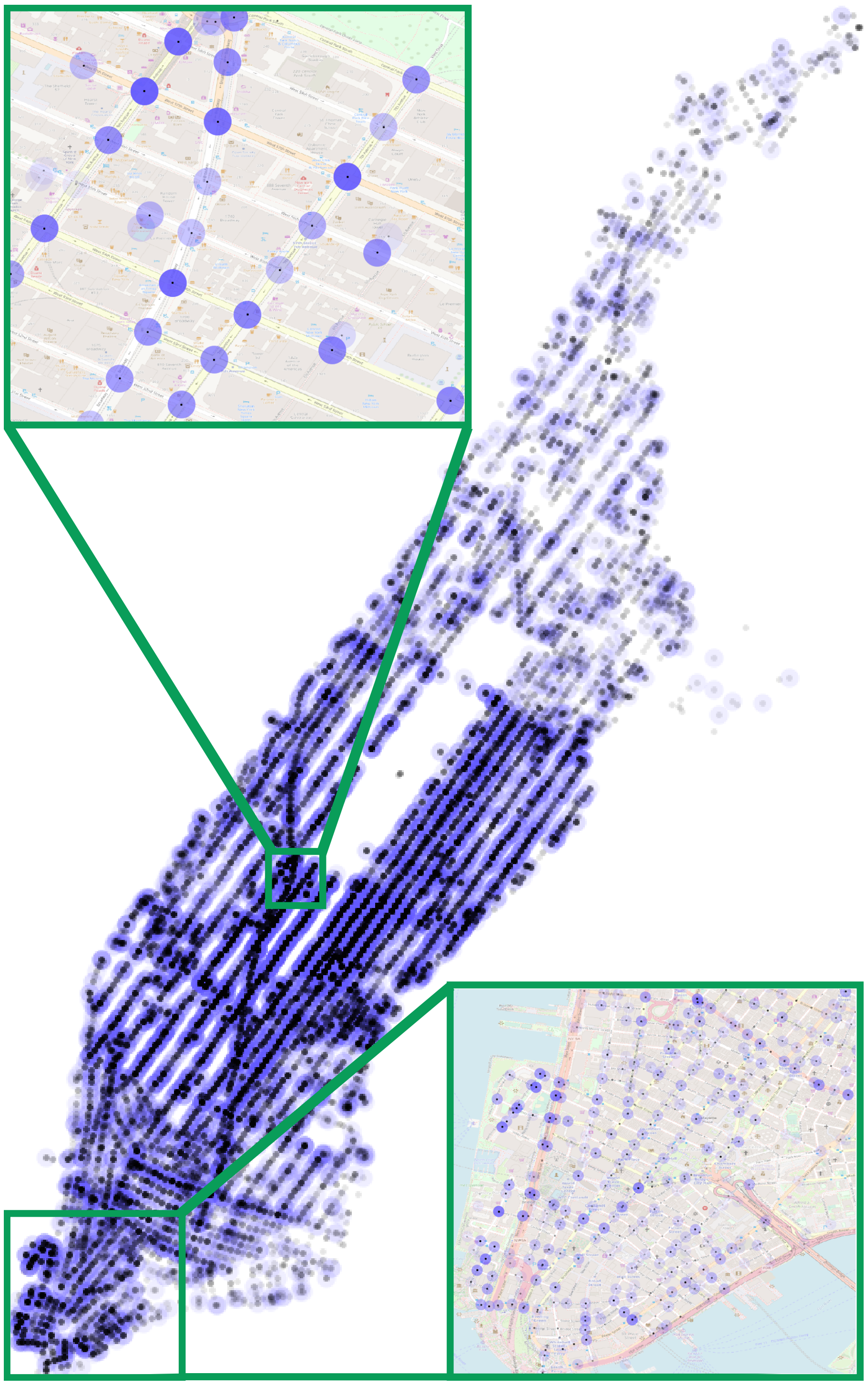


Figure 3.7: Example of a Manhattan instance. The purple areas mark demand: darker color translates to higher travel demand from the area. The Black circles mark vehicles at their initial positions. This particular instance contains 10 362 requests spanning 30 minutes

Table 3.4: Parameters for instance generation

Start time	2022-04-05 18:00:00 (07:00:00)[a]
Duration [min]	0.5, 1, 2, 5, 15, 30, 120, 960
Maximum delay ($\delta_{\max}$) [min]	3, 5, 10
Vehicle capacity (c) [persons]	4
Location	NYC, Chicago, Manhattan, DC

[a] For the longest 960 min instances

As the existing MoD DARP instances are not standardized nor publicly available, we cannot compare the proposed instances with them. However, we can compare the proposed instances with the classical DARP instances: this comparison is in Table 3.5.

3.2.3 Experiments

We run a series of experiments to showcase the potential usage of our instances. These experiments use two DARP solution methods to solve DARP problems defined by our instances.

Solution Methods

We evaluated two existing methods for solving the DARP problem:

- the well-known Insertion Heuristic (IH),
- and an optimal solution method, the Vehicle-group Assignment method (VGA) (Alonso-Mora et al., 2017).

IH is a standard heuristic for vehicle routing problems, including DARP. Because of its good tradeoff between solution quality and computational requirements, it is frequently used as a default or baseline method (Bischoff et al., 2017; Campbell & Savelsbergh, 2004; Fiedler et al., 2018; Kalina et al., 2015). Moreover, many metaheuristic methods use it to compute the initial solution (Muelas et al., 2013). The VGA method is an optimal solution method. Based on the existing research, one cannot expect it to solve all instances, especially those with large maximum time delays (Fiedler, Čertický, Alonso-Mora, et al., 2022).

Implementation and Configuration

We implemented both solution methods in C++ using Gurobi solver[9] to compute optimal vehicle-group assignments in the VGA method. We set the time limit for each experiment to 24 h and ran each on AMD EPYC 7543 CPU with between 1 GB and 300 GB of memory and between 1 and 32 threads, depending on the location and solution method (VGA multi-threaded, IH single thread).

[9]State-of-the-art commercial MILP solver, https://www.gurobi.com/

Table 3.5: Comparison of selected proposed instances and classical DARP instances

Instances	**Instance Horizon (τ)**	**Requests**	**Vehicles**	**Mean Trip Dur. ± SD [min]**	**Time Window (ω) [min]**
DC	30 s	4	18	25.0±7.5	3, 5, 10[c]
DC	15 min	163	121	15.8±8.7	3, 5, 10[c]
DC	16 h	3297	218	16.1±8.3	3, 5, 10[c]
Chicago	30 s	5	7	7.3±4.0	3, 5, 10[c]
Chicago	15 min	274	198	13.4±15.1	3, 5, 10[c]
Chicago	16 h	3794	388	17.8±17.9	3, 5, 10[c]
Man.	30 s	140	124	7.9±3.7	3, 5, 10[c]
Man.	15 min	5113	1672	7.7±3.8	3, 5, 10[c]
Man.	16 h	90 533	2011	8.0±4.0	3, 5, 10[c]
NYC	3 s	329	336	10.3±6.2	3, 5, 10[c]
NYC	15 min	10 567	5085	9.9±6.6	3, 5, 10[c]
NYC	16 h	198 727	6461	10.5±6.7	3, 5, 10[c]
classic A[a]	24 h	16-96	2-8	10.6±4.9	15 min[d]
classic B[b]	24 h	24-144	3-13	6.8±3.6	15-45 min[d] or 30-90 min[d] or 15-90 min[d]

[a] (Cordeau, 2006), 48 instances
[b] (Kirchler & Wolfler Calvo, 2013), 20 instances
[c] one instance for each
[d] Combined with full 24 h time windows on same instances, e.g. 15 min window for pickup and 24 h for drop off of the same request.

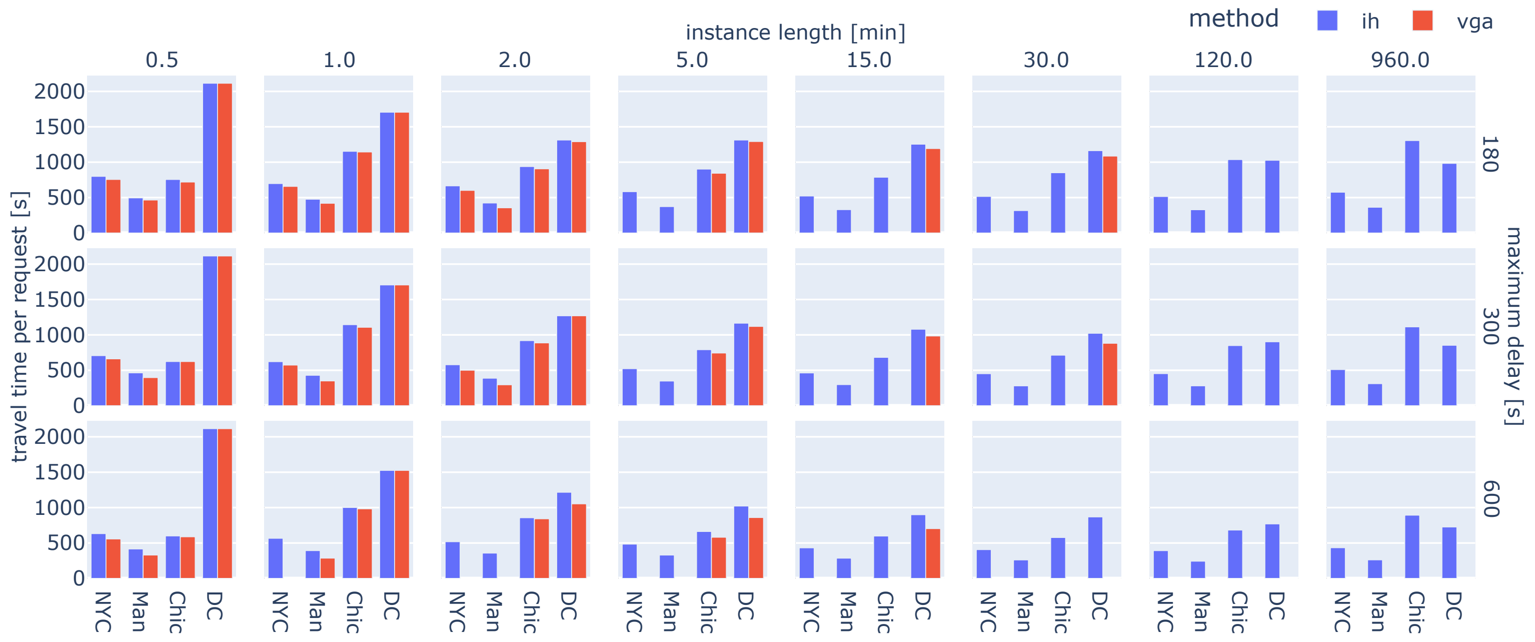


Figure 3.8: Average vehicle travel time per request.

We assume that travel requests are fixed and independent of the travel delay imposed by the solution methods, and all vehicles are operational for the whole time frame of the instance.

Results

We ran both methods on all instances. You can inspect the full results as a part of the instance repository[10]. The main results showing the average cost (vehicle travel time) per travel request are in Figure 3.8. As expected, the VGA method could not solve all instances within the time limit. Another trend is that the cost decreases with the increasing max allowed delay. The same is true for the instance duration. All these trends correspond to findings from previous works (Alonso-Mora et al., 2017; Fiedler, Čertický, Alonso-Mora, et al., 2022).

A more interesting comparison is that between different areas. We can see that area is a more significant predictor of cost than instance duration or max delay. One can argue that this is just an effect of different demand densities but a quick look at Table 3.2 does not support this simple conclusion. The Washington DC instances have greater average costs than the Chicago instances, while the demand density is four times greater. When we compare the costs from New York and Manhattan instances, the less dense New York area indeed shows higher costs. However, the cost difference is not that high if we consider that the demand density is more than ten times lower in NYC. Both these comparisons suggest

[10]https://github.com/aicenter/Ridesharing_DARP_instances

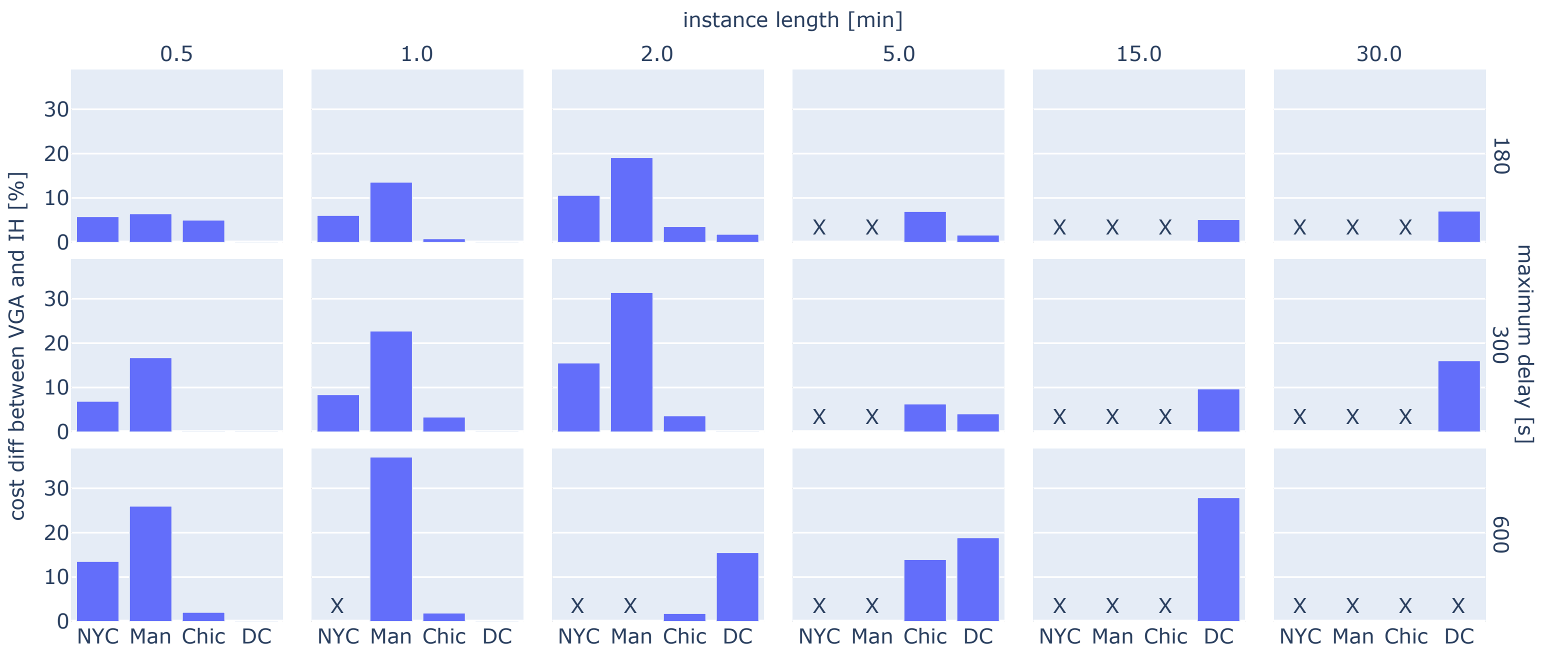


Figure 3.9: The increase of cost when solving by a suboptimal method (IH) relative to the optimal solution (VGA). Instances with duration 2 h and 16 h are omitted. The crosses signalize missing values (missing results of the VGA method).

that there is no simple relation between the demand density and solution cost. Most likely, other more complex area properties, like those examined in Figure 3.3, affect the solution quality as well.

Another important question is how the area affects the cost of the heuristic solution compared to the optimal one. We can see that the difference is not consistent between areas. Figure 3.9 brings more insight into this problem showing the relative difference between the optimal and the heuristic solution for each instance. Again, we can see that the difference between optimal and heuristic solutions increases with instance flexibility (greater duration and maximum delay). Nevertheless, the differences among areas are even greater, suggesting that area-specific properties are an important factor in determining the performance of heuristic methods.

For the final comparison, we show the histogram of occupancies on selected scenarios in Figure 3.10. Again, the occupancy is more impacted by the area than by the other parameters. Manhattan has the highest average occupancy, and the lowest is in Washington DC instances. But the instance length also has a considerable impact on occupancy. We can observe that the occupancy is low in the short instances, most likely due to the inability to match multiple requests to a single vehicle in such a short time.

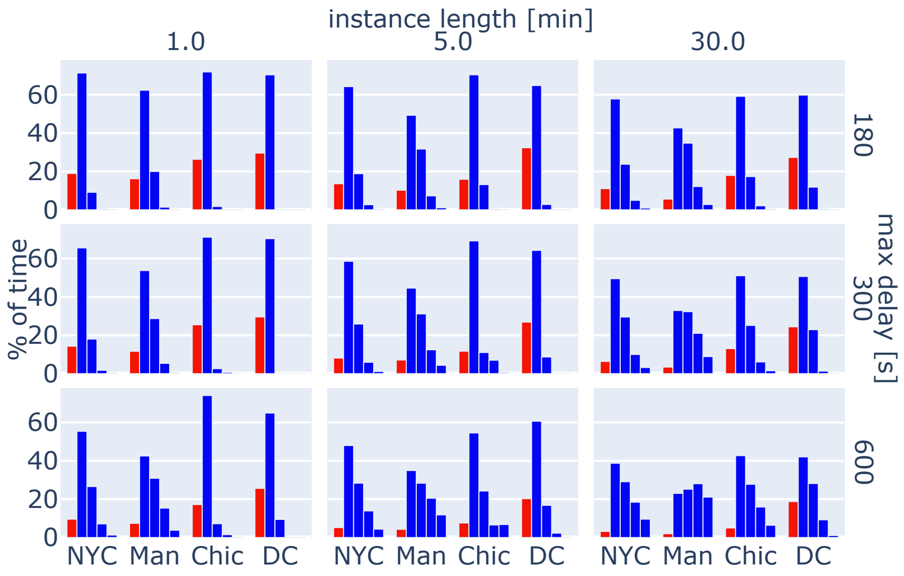


Figure 3.10: Occupancy histogram on selected instances when the problem is solved by the suboptimal method (IH). The red bars mark zero occupancy (empty vehicles), and the blue then start from 1 on the left to 4 (vehicle capacity) on the right.

3.3 Summary

In vehicle routing problems with time windows, and specifically dial-a-ride problems (DARP), significant efforts are dedicated to developing new methods that strike a better trade-off between solution cost and computational requirements. With the emergence of new technologies and the digitalization of transportation, these problems have gained relevance in the context of large-scale mobility-on-demand (MoD) systems. These systems are operated by transportation network companies, some of which already offer ride-sharing options. This is reflected in research where many works analyze the sharability and fleet-sizing and provide DARP solution methods for these systems.

To evaluate these solution methods, we need problem instances containing the travel demand, expected travel times, and vehicles. However, the traditional DARP instances fall short when evaluating methods intended for large-scale MoD systems, even though the problem formulation remains largely identical. Substantial differences in instance characteristics, such as demand density or time window sizes, are the reason. Because of that, works that present solution methods for large-scale DARP usually use different problem instances. Those instances are, however, neither standardized nor publically available. Additionally, most of them are located in the Manhattan area, which can hardly be seen as a representative area due to its abnormally high demand density. To make matters even worse, the obfuscation of the publicly available datasets (including NYC) leads researchers to reuse the datasets released prior to the establishment of the privacy protection measures, which are now about ten years old.

In this chapter, we introduce a comprehensive collection of large-scale instances designed explicitly for evaluating solution methods tailored to large-scale MoD systems with ridesharing capabilities. We have carefully crafted instances with varying characteristics in four areas: New York City, Manhattan, Chicago, and Washington, DC. All instances use actual demand data from these regions, including real travel times for New York City and Manhattan. In total, the dataset contains 96 distinct problem instances. With these instances, new DARP solution methods can be compared with the existing ones without reimplementing them. Moreover, the methodology and implementation described in this chapter can be used to generate additional instances tailored for representing current or future problems.

To showcase the intended usage, we evaluated all instances using two methods, the Insertion Heuristic (IH) and the optimal Vehicle-Group Assignment method. Overall, the results align with findings from previous works: the solution cost per request is decreasing with both the instance duration and the maximum delay for request, and the IH suboptimality is mostly below 20 %. However, an interesting finding is the heavy dependence of all results on the characteristics of the specific areas. Thus, we conclude that evaluating new methods in multiple cities is crucial for methods intended for use within large-scale MoD systems. This contrasts with the current research practice.

Apart from its direct applicability to static DARP, our instances hold great potential for evaluating solution methods addressing online DARP and other MoD-related problems, such as fleet sizing, vehicle depot positioning, pricing, and future demand estimation. Unlike classical DARP instances, which often rely on fictional data, the proposed instances are derived from real-life demand data and road networks and could be used to estimate the real-life performance of DARP solution methods.

CHAPTER 4

Simulation Studies of Large-scale Mobility-on-demand Systems

At the time when this doctoral research was initiated, a lot of research questions related to mobility-on-demand (MoD) were still open. A logical tool to answer these questions and to direct further research was to use traffic simulations. In transportation research, including intelligent transportation systems (ITS), traffic simulation is a well-established and widely used tool (Agatz et al., 2011; Balac et al., 2019; Bischoff et al., 2017; Engelhardt et al., 2019; Hörl et al., 2019; Oke et al., 2020; R. Zhang et al., 2016a). The main advantage of traffic simulation is that it can measure various quantities of interest, without a prior knowledge of the relations in complex traffic systems. With a simulation, it is enough to define the (transportation) entities and their behavior and interactions to analyze the behavior of the whole system and obtain the answers to the research questions.

In this chapter, we demonstrate the use of traffic simulation to answer the research questions related to MoD systems. These questions are presented in Section 4.1. Then we describe the methodology used to answer these questions in Section 4.2. Finally, we present the results of each research question in Section 4.3.

This chapter is based on the journal article:

Fiedler, D., Čertický, M., Alonso-Mora, J., Pěchouček, M., & Čáp, M. (2022). Large-scale online ridesharing: The effect of assignment optimality on system performance. *Journal of Intelligent Transportation Systems*, *0*(0), 1–22. https://doi.org/10.1080/15472450.2022.2121651

and the following conference papers:

(Fiedler et al., 2017) Fiedler, D., Čáp, M., & Čertický, M. (2017). Impact of mobility-on-demand on traffic congestion: Simulation-based study. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC.2017.8317830

(Fiedler et al., 2018) Fiedler, D., Čertický, M., Alonso-Mora, J., & Čáp, M. (2018). The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1173–1178. https://doi.org/10.1109/ITSC.2018.8569451

For all author's publications together with the citation counts and research contributions, see Appendix A.

An important aspect of our work is that we ask our research questions in the context of a hypothetical large-scale deployment of MoD systems. Currently, the MoD systems account for only a small fraction of the total transportation in most cities. However, their share is increasing and for example in Manhattan's central business district, the MoD systems already account for more than 50 % of total transportation (NYC Department of Transportation, 2019) Therefore, in our simulation studies, we consider a complete replacement of all private cars by MoD vehicles. Another important aspect of our work is that we simulate the MoD systems in an online setting, i.e., the travel requests appear continuously. For users of the system, this means that they can request a ride at any time and expect to be served within a reasonable time (limited by the maximum allowed delay $\delta_{\max}$). We have chosen this setting because it represents most of the current MoD systems, provided by TNCs like Uber or Lyft. For the dispatching method, dealing with online ridesharing translates into solving dynamic DARP.

Two other clarifying notes are in order. First, we note that MoD systems are ultimately envisioned also to include high-capacity transportation modes (e.g., trains, subway, buses) and to allow for transfers between different vehicles (Susan Shaheen & Adam Cohen, 2020). However, we focus on MoD systems that transport each passenger from their pick-up to their destination in one vehicle. Second, we focus on MoD systems, where each vehicle is driven by a for-hire driver who transports travelers between their desired pick-up and drop-off locations. Alternatively, in the future, these vehicles may be self-driving. Apart from that, there is a distinct concept called *peer-to-peer ridesharing*, where the vehicle is typically owned and driven by one of the travelers, whose primary motivation is to reach his/her intended destination. Readers interested in peer-to-peer ridesharing are referred to the growing body of research devoted to this model, for example, the work of M. Li et al. (2020) that studies the impact of high occupancy toll lane configurations on the willingness of (peer) drivers to share a ride, or J. Ma et al. (2020) and Yan et al. (2019) who study the ridesharing user equilibrium in the context of peer-to-peer ridesharing.

4.1 Research Questions

In this section, we present the three research questions that we have answered using traffic simulation. Each question was addressed in one of our papers; The papers together with the corresponding research questions and their main contributions are summarized in Table 4.1. For each question, we first present the motivation and importance of the question and then we describe the basic metrics that we use to answer the question.

4.1.1 Research Question 1: What are the impacts of large-scale MoD system deployment?

The main advantage of MoD systems compared to private cars is that they can serve the same travel demand with fewer vehicles since a typical shared vehicle serves more passengers daily than a typical personal vehicle. A combined effect of the reduction of the number of vehicles and their higher utilization has the potential to drastically reduce demand for

Table 4.1: Summary of the research questions. In the second column, there is a reference to the paper that answers the question. In the third column, we list all papers that simulate the particular scenario. This is important as we simulated scenarios related to already answered questions in later papers to provide a comparison with the new scenarios (as the details of the simulation experiments changed between papers).

Research question	**Answered in**	**Addresed in**
What are the impacts of large-scale MoD system deployment?	(Fiedler et al., 2017)	(Fiedler et al., 2017; Fiedler, Čertický, Alonso-Mora, et al., 2022; Fiedler et al., 2018)
Research Question 2: How much ridesharing can improve the performance of large-scale MoD systems?	(Fiedler et al., 2018)	(Fiedler, Čertický, Alonso-Mora, et al., 2022; Fiedler et al., 2018)
Research Question 3: How significant is the gap between optimal ridesharing assignment and simple heuristic?	(Fiedler, Čertický, Alonso-Mora, et al., 2022)	(Fiedler, Čertický, Alonso-Mora, et al., 2022)

urban parking space. however, despite having the clear benefit of reducing the parking space requirements, it is also important to understand the impact of large-scale vehicle sharing on total traffic intensity and traffic congestion. Indeed, to transport all the passengers, shared vehicles need to cover at least the same distance as conventional personal vehicles. However, unlike personal vehicles, shared on-demand vehicles must also travel empty between the individual passengers, i.e., from the destination of the current passenger to the origin of the following passenger. On top of that, passenger demands are not distributed uniformly in time and space, which causes vehicles to accumulate in some parts of the city while other areas may suffer from the shortage of vehicles. To ensure that each region of a city has a sufficient number of available vehicles, the *rebalancing* has to be implemented. Therefore, on-demand transportation increases the total amount of traffic in the system.

To wrap it up, we want to analyze a) the positive effect of vehicle reduction, and b) the negative effect of empty trips between customers and rebalancing trips on total vehicular traffic and traffic congestion.

Existing Research

The properties and performance characteristics of MoD systems have been previously studied either using simulation or by formal analysis of mathematical models of such systems. In (Fagnant & Kockelman, 2014), for example, a multi-agent simulation was used to analyze the impact of deployment of a small fleet of MoD vehicles, and the results suggest that the fleet can be reduced nearly twelve times, but the total distance traveled increases by 10.7 %. The authors also analyze the environmental impacts such as the number of cold starts, finding that the emission savings would be significant. The study by Spieser et al. (2014) considers a replacement of all private vehicles in Singapore by a fleet of shared vehicles.

Their result suggests a more modest reduction of the fleet size: they conclude that approximately one-third of the current number of private vehicles is needed to serve the existing travel demand. Another Singapore study by Marczuk et al. (2015) compares the number of cars needed when using different vehicle rebalancing methods. In results for offline and online rebalancing, 28 % and 23 % fewer cars are needed compared to the system without rebalancing. The idea of replacing all private vehicles with a fleet of shared vehicles was also studied by Bischoff and Maciejewski (2016) in a Berlin simulation scenario. This study suggests that the fleet size can be reduced tenfold from about 1.1 million private vehicles to 90 000 − 100 000 shared vehicles.

The congestion effects of rebalancing in MoD systems have been numerically explored in (R. Zhang et al., 2016a) leading to the following structural insight: Although the need for rebalancing in MoD systems generates a significant amount of extra traffic, the rebalancing vehicles travel in the opposite direction than the vehicles carrying passengers, and thus they use lanes that are currently underutilized. Numerical experiments in simple synthetic networks indeed confirm the observation, suggesting that if the demand traffic flows (vehicles carrying passengers) are below the road capacity limit, then the rebalancing flows (rebalancing vehicles) will also not exceed the capacity and consequently they will not introduce new congestion. Then in (R. Zhang et al., 2016b) the problem of vehicle routing in congested networks was modeled to formally study the impact of rebalancing on formation on congestion. Assuming a symmetric network and time-invariant demand, the authors proved that if the demand flows can be routed without exceeding free-flow capacity, then the rebalancing vehicles can be also routed through the network without exceeding road capacities. Further, they performed experiments demonstrating that the method remains effective even in real-world networks that are not perfectly capacity-symmetric. Also, the authors analyzed the system in a steady state, i.e., they work with the assumption that the travel demand is time-invariant. Clearly, real-world travel demand changes over time (e.g., compare travel demand in the morning peak and the demand at night), but, as the authors point out, if the demand intensities change slowly relative to the average trip duration, the model remains reasonably valid. In large urban areas, however, the traffic intensity may rise drastically over the time of a single trip. Consider, for example, a 30-minute car trip and the difference between traffic intensities at 6:00 and 6:30, i.e., during the onset of the morning peak.

Contribution

We analyze the impact of large-scale MoD system deployment using a multi-agent simulation. Unlike Fagnant and Kockelman (2014) who consider only a small fleet of vehicles, we evaluate a hypothetical large-scale MoD system that replaces all private vehicles in a city. Moreover, we not only quantify the benefits of reducing the number of vehicles like in (Spieser et al., 2014) and (Bischoff & Maciejewski, 2016), but we also quantify the negative effects of empty trips and rebalancing trips on total vehicular traffic and we also analyze the impact on traffic congestion by analyzing traffic density increase. Finally, the road network we use is not capacity symmetric, which can challenge the results by R. Zhang et al. (2016b)

4.1.2 Research Question 2: How much ridesharing can improve the performance of large-scale MoD systems?

In the previous research question, we explained that the deployment of MoD systems reduces the number of vehicles needed to serve the existing travel demand, but at the same time, it increases the total amount of traffic in the system due to empty trips between passengers and rebalancing trips. A possible solution to the increased traffic is to implement large-scale *ridesharing*, i.e., to transport multiple passengers in one vehicle simultaneously. Efficient ridesharing can increase vehicle occupancy and consequently reduce the total driven distance and the required fleet size. The objective of this research question is to quantify the potential of ridesharing to reduce vehicular traffic in a large-scale MoD system. Analyzing ridesharing in the context of on-demand systems is essential because it can answer the question of whether the MoD system can be, in fact, deployed without overloading the capacity of existing road infrastructure.

Existing Research

Ridesharing was traditionally formalized in the framework of Vehicle Routing Problems (VRP), typically as a specific variant of VRP with Pickup and Delivery or, most precisely, a Dial a Ride Problem (DARP) (Cordeau & Laporte, 2007; Toth & Vigo, 2014). Yet, as explained in Section 3.1.2, the existing VRP methods focus on instances with very different characteristics than the instances that appear in large-scale MoD systems.

The potential for large-scale ridesharing within MoD systems was studied using the shareability network model in (Santi et al., 2014). This analysis of taxi trips in Manhattan revealed that up to 80 % of the trips could be pairwise shared such that the travel time is increased by no more than a couple of minutes. The analysis was later extended to other cities (Tachet et al., 2017). The assumption of the maximum of two passengers in a vehicle was later lifted in (Alonso-Mora et al., 2017), where the authors proposed a technique for dynamic assignment of requests to vehicles in the fleet and applied the technique to analyze the potential of ridesharing within NYC taxi dataset.

Contribution

In contrast to the above work by Alonso-Mora et al. (2017) that analyzes the scenario of replacing all taxis in Manhattan with a fleet of 3000 shared taxis, we analyze the scenario of replacing all private vehicles with a fleet of 50 000 shared MoD vehicles and study the impact of large-scale ridesharing. Specifically, we quantify the potential of ridesharing to reduce the total distance traveled, but also traffic load, fleet size, and the delay of passengers.

4.1.3 Research Question 3: How significant is the gap between optimal ridesharing assignment and simple heuristic?

In the last research question, we ask how much more efficient can the MoD system with ridesharing be if the vehicle plans are computed optimally. If the optimal solutions are significantly better compared to the solutions computed by simple heuristics used in the previous research question, it would be reasonable to invest the effort to deploy the optimal solvers in real-world MoD systems. On the other hand, the difference can be negligible, in which case the simpler heuristics would be sufficient. The objective of this research question is to quantify the difference between the optimal and heuristic solutions in terms of the total vehicle distance traveled, and also to measure other properties like the number of used vehicles or the impact on traffic congestion.

Existing Research

Recently, a number of mobility-on-demand system models have been developed with the aim of providing quantitative insights into the potential of large-scale ridesharing to improve the efficiency of urban transportation. The ridesharing problem is commonly formulated as a Vehicle Routing Problem with Pickup and Deliveries (VRPPD) or, more specifically, as Dial-a-Ride Problem (DARP) (Cordeau & Laporte, 2007; Toth & Vigo, 2014). These formulations can be solved optimally using off-the-shelf Integer Linear Programming (ILP) solvers or domain-tailored ILP solution techniques. However, the applicability of these methods is limited to small-scale instances with at most tens of requests and vehicles. Large-scale MoD systems typically require the ability to find routes for many more vehicles and requests. For example, in New York City (NYC), there are almost 100 000 active taxis per hour during peak traffic (NYC Taxi & Limousine Commission, 2018). Therefore, DARP instances appearing in large-scale MoD systems are typically solved using heuristic methods.

A popular heuristic method for large-scale DARP is the Insertion Heuristic, described in Section 2.3.1.

The metaheuristic methods, which are effective in solving conventional DARP problem instances (Ho et al., 2018), typically target scenarios with less than twenty vehicles (Masmoudi et al., 2016; Pfeiffer & Schulz, 2022) and suffer from scalability issues when applied to large-scale DARPs. However, in the last decade, there has been some progress with metaheuristic approaches enabling solving larger instances. Jung et al. (2015) used simulated annealing to solve scenarios with 600 operating vehicles. Another popular metaheuristic is the Greedy Randomized Adaptive Search Procedure (GRASP) used by Santos and Xavier (2013). The authors were able to solve instances with up to 750 requests. They also tested an online setting with 78 000 requests per day, and later, they improved the results significantly (Santos & Xavier, 2015). Muelas et al. (2013) also solved four types of specialized DARP scenarios with up to 90 vehicles using Variable Neighborhood Search. Later, Muelas et al. (2015) modified this approach to a distributed version which was able to solve scenarios with up to 1668 vehicles and 16 000 requests. Another metaheuristic, a modified artificial bee colony algorithm, was used by Zhan et al. (2021). The method was able to solve an instance of 3661 requests and 2400 vehicles. Later, this method was used in a simulation-optimization framework for an MoD system with electric vehicles (Zhan et al., 2022).

Apart from heuristics and metaheuristics, there has been a new research path of developing exact methods for large-scale DARP instances motivated by large MoD systems. A systematic and scalable approach for pairwise ridesharing based on bipartite matching in the so-called shareability network was proposed by Santi et al. (2014). The analysis revealed that up to 80 % of the trips could be pairwise shared while keeping the travel delay lower than a couple of minutes. Later, Alonso-Mora et al. (2017) proposed a new method that lifted the limit of two passengers per car and evaluated this method on the NYC taxi dataset. Finally, Čáp and Alonso-Mora (2018) utilized this method to study the trade-offs between the quality of service and the operation cost inherent in ridesharing.

Contribution

In this research question, we analyze the impact of passenger-vehicle assignment optimality on MoD system performance. To do this, we use a variant of the vehicle-group assignment (VGA) method used by Alonso-Mora et al. (2017) and Čáp and Alonso-Mora (2018). We chose this method because it was previously demonstrated to be able to efficiently solve large-scale DARP instances with tight pick-up and drop-off time windows that are characteristic of current large-scale MoD systems. Moreover, it is less complex than the classical exact methods for DARP, and it can be easily modified to a resource-constrained version, which we also evaluate in this work.

The contribution of this paper is 3-fold:

1) Optimality: We took special care to ensure that all ridesharing assignments and routes are computed optimally. This is in contrast to Alonso-Mora et al. (2017), who used a similar solution algorithm to evaluate shareability within the NYC taxi dataset, but to maintain computational tractability, the actual implementation used in the experiment resorted to heuristics and time-outs, leading to suboptimal performance of the system. Moreover, it remained unclear how far are the reported performance metrics from optimum. In this work, we identified and solved several algorithmic bottlenecks, and consequently, we were able to obtain optimal ridesharing assignments for the majority of the evaluated scenarios.

2) Scale: We implemented performance optimizations that enable us to significantly scale the algorithm and compute optimal ridesharing assignments for instances of unprecedented size peaking at more than 21 000 active travel requests and 11 000 vehicles. This is in contrast to Čáp and Alonso-Mora (2018) who proposed the optimal version of the VGA method but were only able to solve problem instances with a bit less than 500 requests.

3) Impact of Assignment Optimality: With an 1) *optimal* and 2) *scalable* implementation of a ridesharing algorithm, we are able to achieve the main objective of this work: to quantify the impact of using an optimal ridesharing method on system performance in comparison to the performance achieved by sub-optimal ridesharing methods. We quantify the reduction in vehicle distance traveled, travel delay, used vehicles, and *traffic density* (vehicles per meter of the road) for different ridesharing strategies. This allows us to give a quantitative answer to the question of how much we gain by actually making the effort to compute optimal assignments.

4.2 Methodology

To answer all three research questions, we use multi-agent simulation. The methodology of our simulation experiments can be divided into three parts: a) the simulation inputs,

b) the MoD system design, and c) the simulation itself. The demand data and the travel time model we use are described in Section 4.2.1. Next in Section 4.2.2, we present the evaluated MoD system design. Finally, in Section 4.2.3, we describe the simulation tool and the simulation process itself.

4.2.1 Input Data

Our simulation research preceded the development of large-scale DARP instances based on historical data, described in Section 3.2. Because of this, we had to create our own instances based on other data sources.

For all our simulation studies, we use the synthetic demand for the city of Prague, the Czech Republic, generated by the multi-agent activity-based model described in Drchal et al. (2019). We used the Prague demand because a) we have access to the synthetic travel demand model for the area and b) because its demand density, demand structure, and road topology are representative of a large European city. This is in contrast to previously considered urban areas, such as Manhattan or Singapore, which due to an extremely high density of travel demand, lead to overly optimistic estimates of the benefits of ridesharing. The details of the Prague travel demand and the travel time model are described in the next section (4.2.1). To provide at least partially comparable results to other studies, we also extended our simulation research to the Manhattan area in the last study (Fiedler, Čertický, Alonso-Mora, et al., 2022). We describe the Manhattan demand and travel time model in Section 4.2.1.

As for the road network, we use the road networks extracted from OpenStreetMap[1] to cover the area of interest. For Prague, the network consists of 158 674 edges and 63 995 nodes, for Manhattan, the network consists of 9676 edges and 4467 nodes.

Prague Demand and Travel Time Model

For Prague, we used the synthetic travel demand generated by a multi-agent activity-based model described in Drchal et al. (2019) (see Section 3.1.3 for context on the generated demand). The model covers a typical workday in the metropolitan area of Prague. The population of over 1.3 million is modeled by the same number of autonomous, self-interested agents, whose behavior is influenced by their sociodemographic attributes, current needs, and situational context. Individual decisions of the agents are implemented using four modules responsible for choosing the activity type, duration, location, and mode. Each module uses a dedicated machine learning model (such as neural network, decision trees, regression tree) trained so that its output matches various real-world data sets such as travel diaries and other transportation-related surveys, demographic data, points of interest, and transport network structure. Planned activity schedules are simulated and tuned, and finally, their temporal, spatial, and structural properties are validated against additional historical real-world data (origin-destination matrices and surveys) using the six-step validation framework VALFRAM (Drchal et al., 2016).

[1] https://www.openstreetmap.org/

The model consists of over three million trips by all modes of transport in one 24-hour scenario, out of which there are roughly one million trips realized by private vehicles (Figure 4.1). Tables 4.2, 4.3, and Figure 4.2 show example activity schedules for two agents. In this work, we select only the trips realized by private vehicles in two representative time intervals: the *peak* dataset includes trips that start at 06:30 and 08:00, and the *off-peak* dataset includes trips that start between 10:30 and 12:00. The two datasets contain about 130 000 and 45 000 trips, respectively. The duration of trips ranges from 1 to 37 minutes; the histogram is in Figure 4.3.

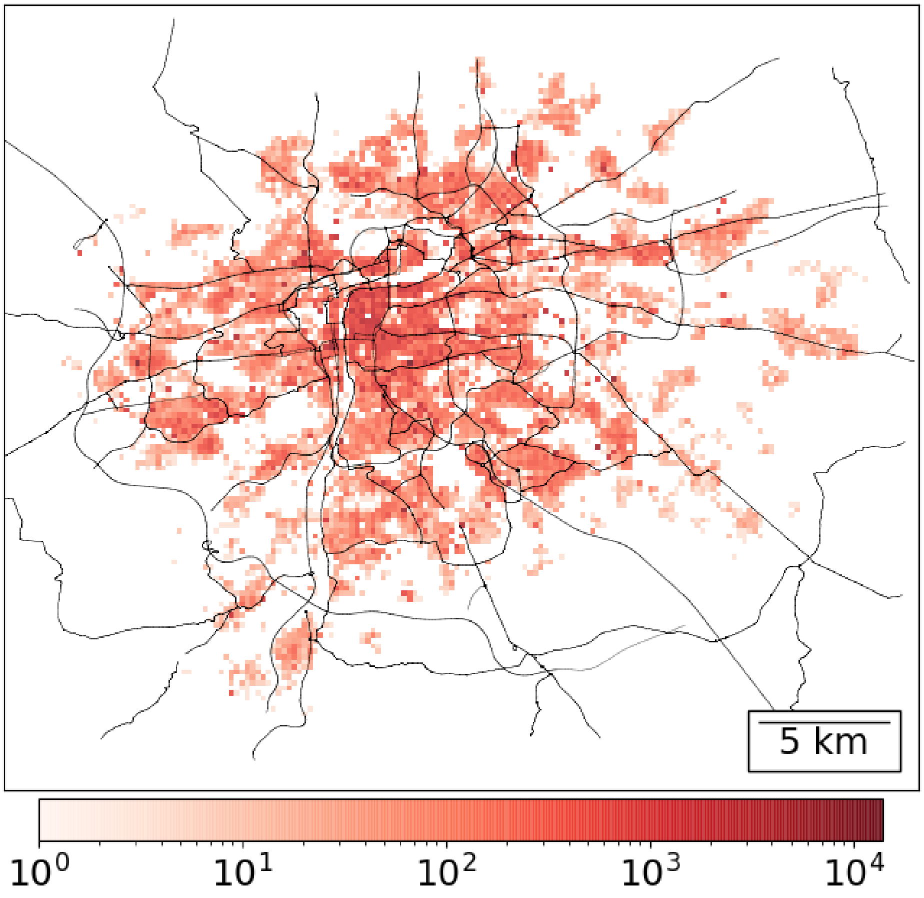


Figure 4.1: Demand for personal vehicle traffic in Prague. The start positions of all vehicle trips are discretized to squares of 200 square meters. A darker color translates to higher demand, and the color bar has a logarithmic scale.

The travel time model is based on the maximum speed limits on each road segment, i.e., the vehicles travel at the maximum speed allowed by the speed limit. The speed limit for each road segment was taken from OpenStreetMap data, and missing entries were generated according to the following rules based on the local legislation: highway: $130\,\mathrm{km\,h^{-1}}$, living street: $20\,\mathrm{km\,h^{-1}}$, otherwise: $50\,\mathrm{km\,h^{-1}}$.

Table 4.2: Example trips. Each trip connects two activities, shown in Table 4.3. We can identify each activity by `Person` column and `From`/`To` columns that correspond to the `Activity` column in Table 4.3

Trip	Person	From	To	Mode
4 500 942	50 719	0	1	PT
4 500 943	50 719	1	2	PT
4 500 944	50 719	2	3	WALK
4 500 945	50 719	3	4	PT
4 789 903	450 277	0	1	CAR
4 789 904	450 277	1	2	CAR
4 789 905	450 277	2	3	CAR

Table 4.3: Example activities.

Person	Activity	Start	End	Type	Lat	Lon
50 719	0	00:00	04:06	SLEEP	50.084 294	14.490 635
50 719	1	06:56	07:42	LEISURE	50.110 286	14.496 852
50 719	2	10:31	13:49	WORK	50.086 623	14.461 201
50 719	3	15:01	16:04	LEISURE	50.076 027	14.439 032
50 719	4	17:17	00:00	SLEEP	50.084 294	14.490 635
450 277	0	00:00	07:22	SLEEP	50.131 751	14.423 139
450 277	1	07:54	15:43	WORK	50.084 170	14.360 924
450 277	2	16:00	16:35	SHOP_LONG	50.059 205	14.420 547
450 277	3	17:26	00:00	SLEEP	50.131 751	14.423 139

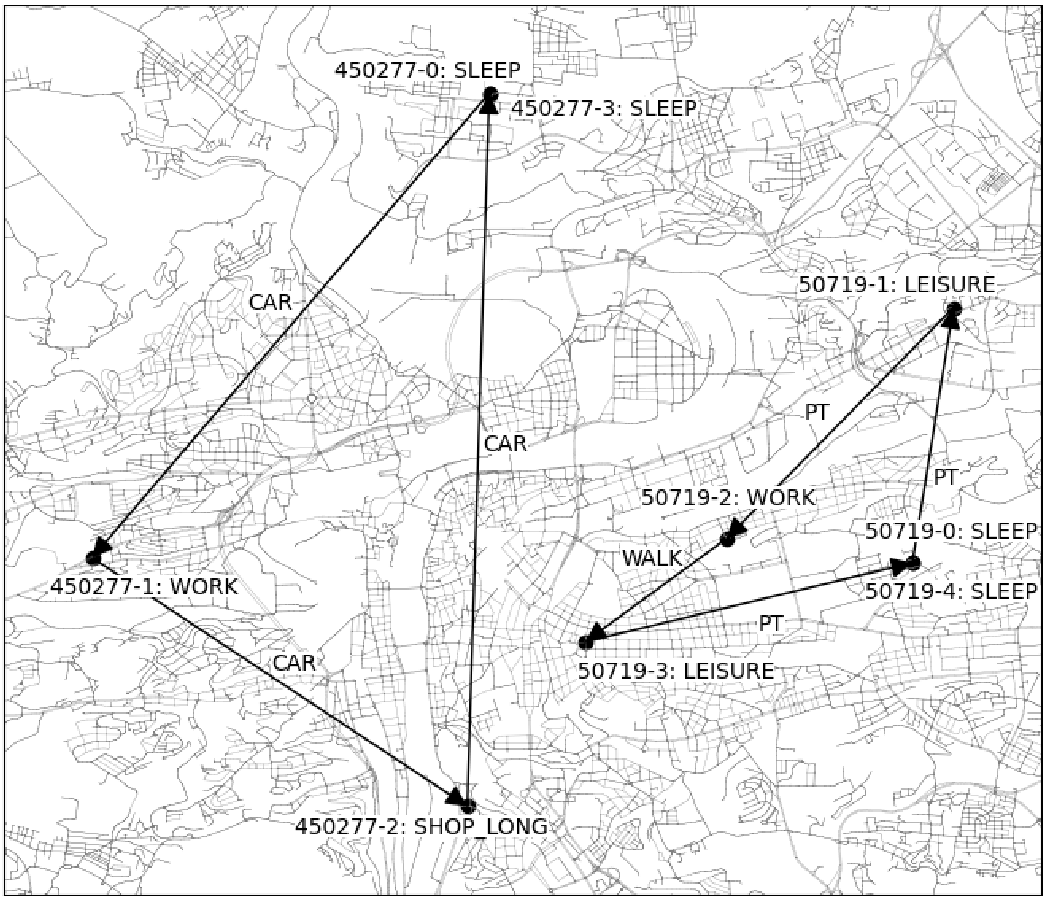


Figure 4.2: Two example trips from the generated demand. The filled circles represent activities, while the arrows represent trips between those activities. Next to each activity, we can see the person and activity IDs in the format `person_id-activity_id`. The activities corresponding to these IDs can be found in Table 4.3.

Manhattan Demand and Travel Time Model

In the last study, we also experimented with the Manhattan area. For that, we used the Manhattan taxi demand dataset a favorite dataset in the transportation community. Specifically, we use the same demand and road network as used by Alonso-Mora et al. (2017). We tried to mimic the setup of Alonso-Mora et al. (2017) in order to provide comparable results, as this work presents the method that we use as an optimal solution method for our DARP instances. It may seem superfluous to evaluate the same method on the same dataset again. However, as explained in Section 4.2.2, the Alonso-Mora et al. (2017) used a relaxed version of the method in the experiment.

Identically to our Prague experiment, we simulated the system for one hour with a 30-minute warm-up period. While Alonso-Mora et al. (2017) run the simulation for one week's worth of data, here, for simplicity, we selected the day and hour with the largest number of requests, which was Friday, May 10, 2013, between 19:00 and 20:00. There are 137 202

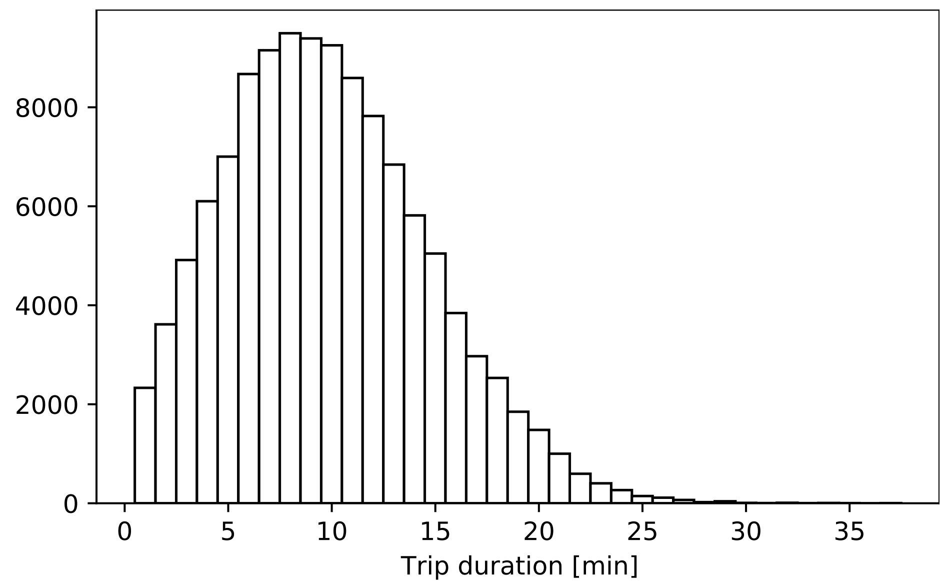


Figure 4.3: Histogram of fastest path travel times for each trip.

travel requests in the selected period, Figure 4.4 shows the spatial structure of the demand.

Like Alonso-Mora et al. (2017), we use travel speeds along individual road segments derived from historical data. However, instead of computing the speeds from the travel demand, we use the speeds from the Uber Movement[2] open data project.

4.2.2 MoD System Design

For MoD systems design, we adopt a station-based methodology described by Pavone et al. (2012) or by Wallar et al. (2019a), which means that idle MoD vehicles are parked in dedicated parking facilities instead of parking on-street or cruising. This setup is typical in carsharing or bike-sharing systems because curb parking would take valuable urban space, and cruising for parking would increase fuel consumption and congestion. Further, in the case of electric vehicles, stations will provide charging infrastructure. Finally, the stations can be used as a base for vehicle maintenance and cleaning.

Vehicles are initialized in stations and leave a station only to serve travel requests. Whenever a vehicle becomes idle, it starts driving to the nearest station to park there, till it is dispatched again.

The dispatching of vehicles to serve travel requests is realized by one of the developed dispatching methods, these are described in Section 4.2.2. To distribute the stations in the road network, we use the station positioning method described in Section 4.2.2. To ensure that there are always enough vehicles in each station, we use a rebalancing method that continuously redistributes vehicles from stations with a surplus of vehicles to stations with a shortage of vehicles. This is described in Section 4.2.2. Finally, it was necessary to determine the number of vehicles in each station, this is described in Section 4.2.2.

[2]https://movement.uber.com/

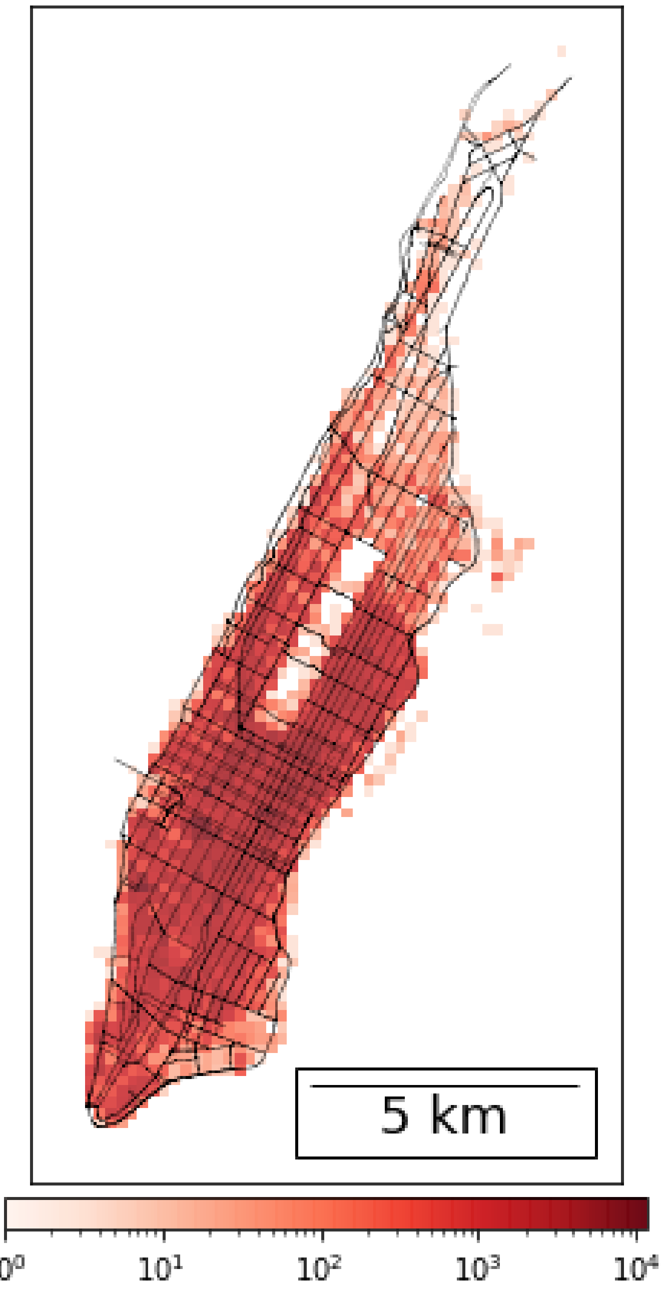


Figure 4.4: Manhattan taxi trip requests on Friday, May 10, 2013, between 19:00 and 20:00. The start positions of all vehicle trips are discretized to squares of 200 square meters. A darker color translates to higher demand, and the color bar has a logarithmic scale.

Dispatching

In an MoD system, new requests dynamically arrive and need to be served. When planning how the users of the MoD system should be served by the vehicles, we desire to minimize the total operational cost of the system, such that the discomfort of every passenger is bounded by $\delta_{\max}$. This translates to solving the dynamic DARP problem. We deal with the dynamic aspect by running DARP solution methods periodically, we refer to such planning period as a *batch*. During one batch, we collect all newly announced requests and execute a planning procedure that computes request-vehicle matching and corresponding vehicle plans. We make the following assumptions: a) travel time on each road segment is constant over time and does not depend on the number of vehicles on the segment, b) the execution of the vehicle schedule is perfect (there are no random delays), and c) the mode choice is fixed in the demand model and customers accept any plan that satisfies the max delay constraint (which is guaranteed in our setup, as explained in Section 4.2.2).

We implemented and compared two methods for solving dynamic DARP. The first one is the Insertion Heuristic (IH), and the second one is the Vehicle-Group Assignment (VGA) method. Both methods are described in Section 2.3. In the remainder of this section, we will describe the modifications we have made to those two methods in order to work with dynamic DARP and also, we present two heuristic variants of the VGA method we have evaluated.

To adapt the IH to dynamic DARP, we simply start with the set of existing plans for each vehicle instead of empty plans, like it is described in Section 2.3. For the VGA method, we need to modify the group generation procedure. Here the modifications are more complex: we need to keep track of what generated groups contain all onboard requests and consider only those groups in the assignment ILP. The modified group generation procedure is described in Algorithm 4.

Finally, we also evaluated two heuristic variants of the VGA method. While the ability to compute optimal ridesharing assignments is essential to understanding the limit of performance gains that can be achieved by ridesharing (and the gap between the optimal performance and the performance of the heuristic solutions), a practical deployment may impose constraints on the maximum run time of a ridesharing algorithm. Therefore, we also tested two resource-constrained versions of the VGA method. The first one limits the ILP solver optimality gap set to 0.5 % and the group generation time to 60 ms per vehicle. We refer to this version as *VGA limited*. The second one is a reimplementation of the method proposed by Alonso-Mora et al. (2017). We refer to this version as *VGA PNAS*.

Station Positioning

The vehicle stations were generated such that every location on the road network (excluding roads without travel requests such as tunnels or highways) can be reached from one of the stations within 210 s, and the number of stations is minimized. We compute the station positions using an integer program with binary variables s_l for each location l in the set of serviced locations L_s, where each variable s_l indicates if there is a station at location l (1)

Algorithm 4 Function `generate_groups` described in Section 2.3.2 modified for dynamic DARP. The boolean-valued function $f(R, a)$ evaluates to true if a feasible plan serving all requests in group R exists for vehicle a. The set of temporary groups α_a contains all feasible groups for vehicle a, while the set of final groups γ_a contains only those groups that contain all onboard requests.

```
Input: A vehicle $a$ and the set of requests $D$ and a set of onboard requests $D_a$.
  for $r \in D$ do                                              ▷ generate groups of size 1
      if $f(\{r\}, a)$ then
          $\alpha_a^1 \leftarrow \{\{r\}\} \cup \alpha_a^1$
          if $D_a \subseteq \{r\}$ then
              $\gamma_a^1 \leftarrow \{\{r\}\} \cup \gamma_a^1$
  $k \leftarrow 1$
  while $\alpha_a^k \neq \emptyset$ do                                    ▷ generate remaining groups
      $\alpha_a^{k+1} \leftarrow \emptyset$
      checked $\leftarrow \emptyset$                                  ▷ not check groups repeatedly
      for $R \in \alpha_a^k, \{r\} \in \alpha_a^1$ do
          if $(R \cup \{r\}) \notin$ checked and $\forall R' \subset (R \cup \{r\}), |R'| = k : R' \in \alpha_a^k$ then
              if $f(R \cup \{r\}, a)$ then
                  $R' \leftarrow R \cup \{r\}$
                  $\alpha_a^{k+1} \leftarrow R' \cup \alpha_a^{k+1}$
                  if $D_a \subseteq R'$ then
                      $\gamma_a^{k+1} \leftarrow R' \cup \gamma_a^{k+1}$
          checked $\leftarrow$ checked $\cup R'$
      $k \leftarrow k + 1$
  $\gamma_a \leftarrow \{\emptyset\} \cup \gamma_a^1 \cup \gamma_a^2 \cup \cdots \cup \gamma_a^k$
```

or not (0). We minimize

$$\sum_{l \in L_s} s_l, \tag{4.1}$$

subject to

$$\sum_{l' \in P_l} s_{l'} \geq 1 \quad \forall l \in L_s, \tag{4.2}$$

where P_l is a set of nodes from which l is reachable within 210 s.

Solving this optimization problem for the Prague Instance resulted in 73 stations shown in Figure 4.5. Because the historical speeds from the Uber Movement dataset used for the Manhattan travel time model are on average approximately half of the posted speed and there are a lot of one-way streets in Manhattan, we need 236 stations to provide the required quality of service there, even though Manhattan is about five times smaller than Prague. Figure 4.6 shows the locations of Manhattan stations.

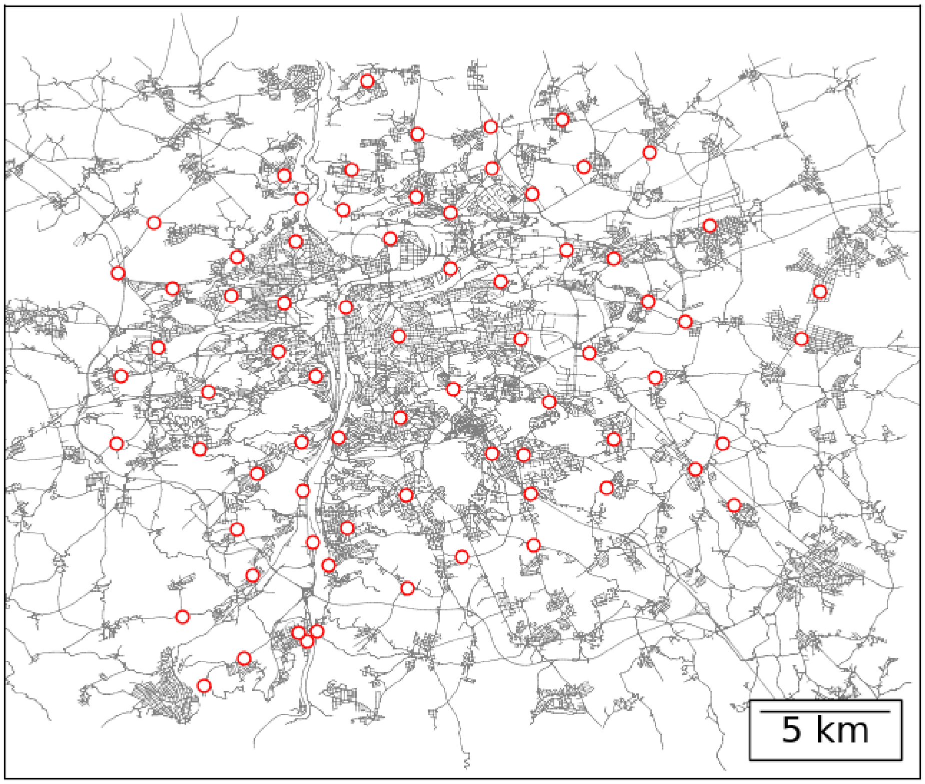


Figure 4.5: MoD system stations in the city of Prague. There are 73 stations in total, shown as red circles.

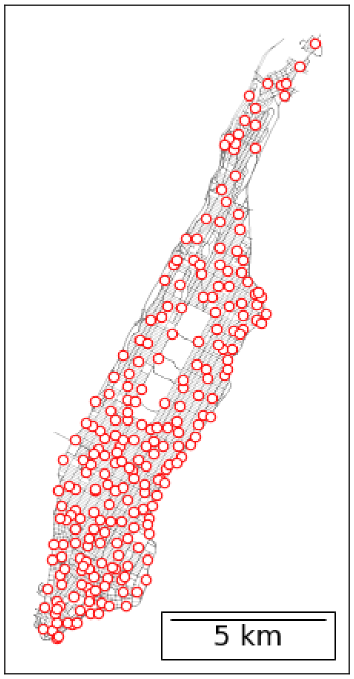


Figure 4.6: MoD stations on Manhattan. Each red circle represents a single MoD system station.

Rebalancing

The stock of vehicles at each station is stabilized by a vehicle rebalancing process that continuously sends empty vehicles from stations with a surplus of vehicles to stations that have a shortage of vehicles. We use the rebalancing policy introduced by Pavone et al. (2012) and later evaluated by Spieser et al. (2014) in the Singapore MoD case study. In one-minute intervals, we generate an integer program for transferring vehicles from stations with more vehicles compared to the initial state to stations with fewer vehicles compared to the initial state such that the number of vehicles in each station s is kept above a corresponding threshold β_s, and the total length of all rebalancing trips is minimized. We experimentally determined that in order to compensate for driving vehicles, the β_s should be no more than 85 % of the initial number of vehicles parked in s. Also, we use only stations with at least 5 % more vehicles over the initial state as source stations in order to prevent rebalancing instabilities, i.e., rebalancing flows in the opposite directions.

Fleet Sizing

Our objective is to achieve full service availability during the entire experiment, i.e., every request should be served. Specifically, to determine the number of vehicles in each station, we first created a dedicated vehicle for each request in the station closest to the requested pickup location. Then, we started iteratively reducing the number of vehicles by the same factor in each station until the first vehicle shortage event occurred in any station. Then, we used the vehicle counts from the last iteration without any shortage. This procedure guarantees that there is a sufficient number of vehicles to serve all requests from the nearest station. We experimentally determined that in order to be able to serve every request during the morning peak without ridesharing, the MoD system requires a total of 68 201 vehicles[3]. We used the same fleet size for other scenarios (off-peak, ridesharing). In practice, we can use a lot fewer vehicles, especially for the ridesharing scenarios. However, since the fleet-sizing problem is not the focus of this article, we used this fleet-sizing method to ensure that the size of the fleet is not the limiting factor. The number of vehicles that were actually used in each experiment is in our experimental results.

4.2.3 Simulation

To answer the research questions given the inputs described in Section 4.2.1 and the system design described in Section 4.2.2, we use the transportation simulation software stack displayed in Figure 4.7. All listed tools Except Gurobi solver are written in Java and licensed under the LGPL license. We have chosen this software stack because it a) handles large-scale simulations containing tens of thousands of agents, and, at the same time b) it can be easily extended not only by its inputs and configuration parameters but also by customizing the behavior of agents.

The simulation itself runs in Alite[4], a framework for multi-agent simulations with an event-based simulation engine (Komenda et al., 2013). The simulation tools and entities for urban mobility are part of the AgentPolis[5]. Agentpolis is a multi-agent event-based transportation simulation framework (Jakob et al., 2012). For the MoD experiments, we developed an extension of AgentPolis called SiMoD (Simulation of Mobility-on-Demand)[6]. In this extension package, we implemented the IH and the VGA method. Also, rebalancing and fleet-sizing are developed in SiMoD. Most parts of SiMoD are implemented in Java, some parts are implemented in Python (e.g., station positioning). The ILPs used in the station positioning, rebalancing and inside the VGA method are solved using Gurobi [7] (Gurobi Optimization, LLC, 2023). Finally, AgenPolis uses the Grap Importer[8] library to import the road network and the Geographtools [9] library for the geography data structures and operations.

In the simulation, every travel request is represented by a passenger agent, and every vehicle is represented by an on-demand vehicle agent. Every crossroad has a corresponding road network node in the simulation, and every road segment has a corresponding edge. At every time, each agent in the simulation has an exact position on the road network. The execution of the simulation is perfect, e.g., there are no random delays. This includes traffic congestion, which is not simulated. In other words, when we talk about congestion

[3]The number of vehicles is smaller than the number of trips. This is possible because even without ridesharing, one vehicle can serve more travel requests sequentially.

[4]https://github.com/aicenter/alite

[5]https://github.com/aicenter/agentpolis

[6]https://github.com/aicenter/simod

[7]http://www.gurobi.com/

[8]https://github.com/aicenter/graph-importer

[9]https://github.com/aicenter/geographtools

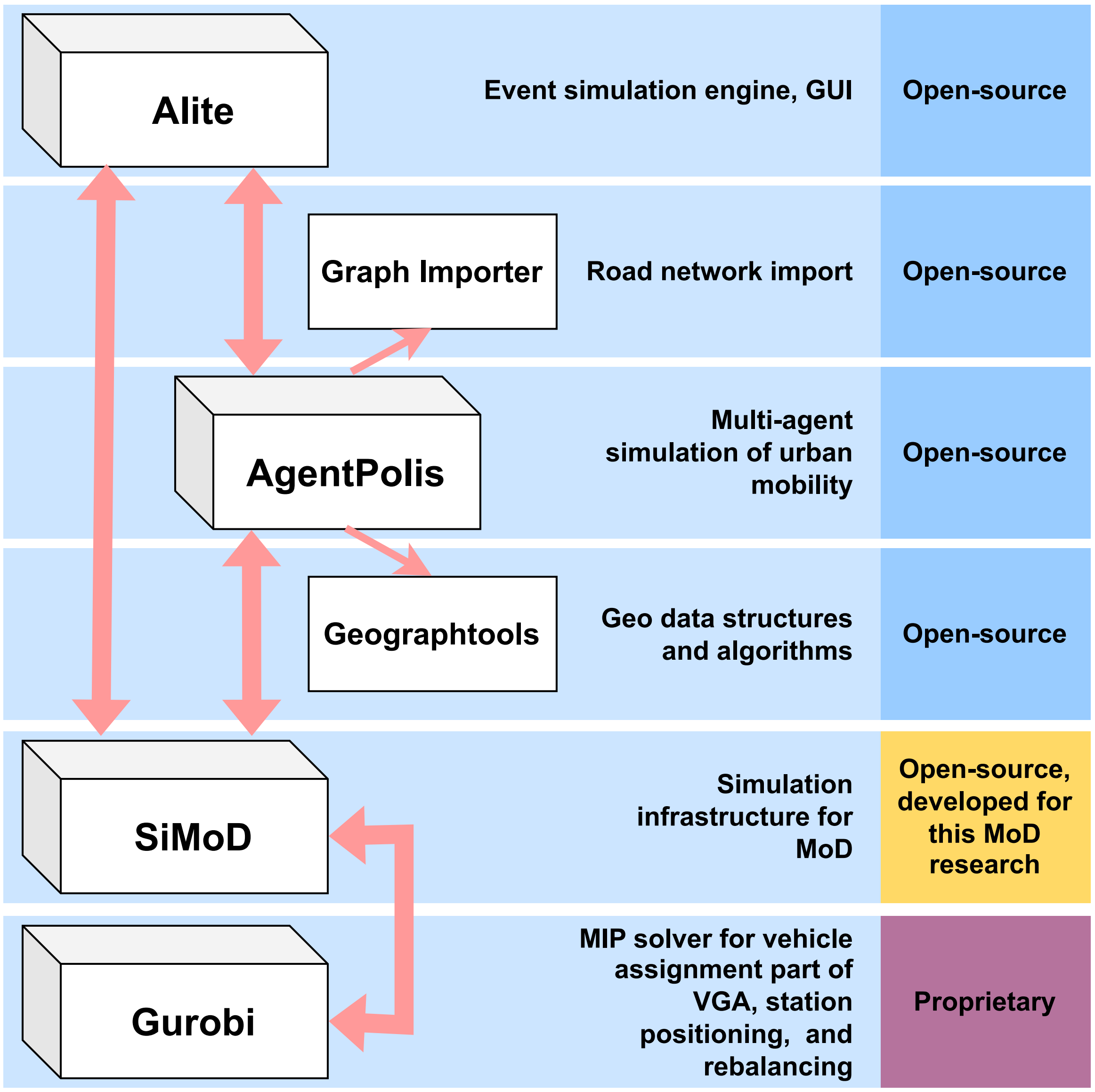

Alite
Event simulation engine, GUI
Open-source
Graph Importer
Road network import
Open-source
AgentPolis
Multi-agent simulation of urban mobility
Open-source
Geographtools
Geo data structures and algorithms
Open-source
SiMoD
Simulation infrastructure for MoD
Open-source, developed for this MoD research
Gurobi
MIP solver for vehicle assignment part of VGA, station positioning, and rebalancing
Proprietary

Figure 4.7: The simulation software stack.

in the results, we mean the observed congestion in the form of traffic density on each road segment. However, there are no interactions between the vehicles and no driving model. This simplification has a strong consequence: the (traffic peak) travel times are not realistic, and the simulation results do not represent a real-world deployment. However, we believe that to answer the research questions, this simplification is acceptable, as all research questions are related to the relative performance of the evaluated scenarios. These relative differences can be, of course, also affected by the simplifications. But, as we will show in Section 4.3, the differences are so significant that we believe that the conclusions cannot be affected by the missing congestion modeling. The dispatching procedure runs together with the simulation sequentially, and thus from a simulation perspective, the vehicle plans computation is an instantaneous event. In the case of practical deployment, one needs to achieve sufficiently low wall-clock running time by computing on a computational cluster with many CPU cores.

Simulation Environment, Events, and Lifecycle of the Agents

The simulation environment consists of a) a road network composed of *nodes* (crossroads) and *edges* (road segments), b) vehicle stations, c) on-demand vehicle agents, and d) passenger agents. In Figure 4.8, we show a screenshot of the AgentPolis visualization captured during one of the simulation experiments.

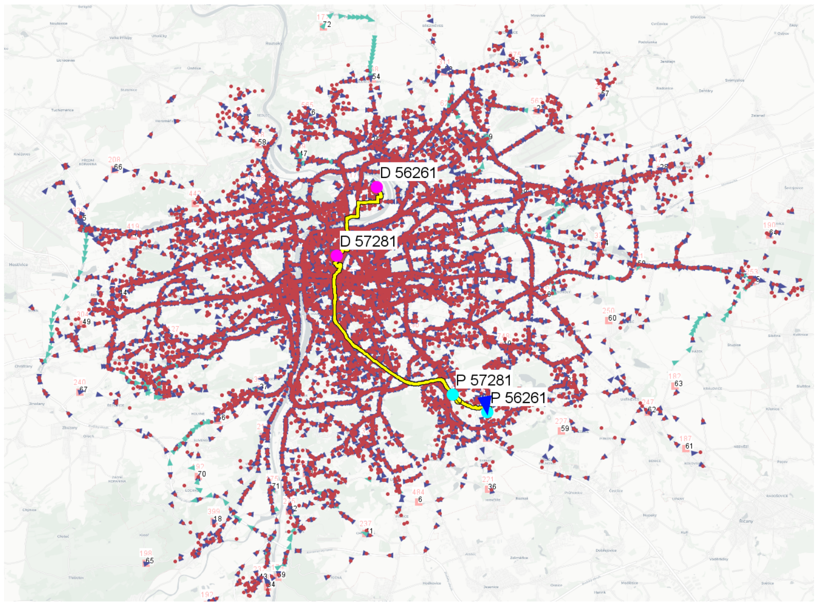
The simulation events can be described as follows. During initialization, the simulation creates vehicle stations, each filled with the pre-determined number of vehicles (see Section 4.2.2). Then, as the simulation runs, the simulation creates passenger agents for each request at its announcement time and origin point. Periodically, the simulation executes the dispatching procedure (see Section 4.2.2), which assigns passengers to vehicles and computes the corresponding vehicle plans. Note that each passenger can be either matched to one of the empty vehicles parked in a station or to a vehicle already serving some previously assigned requests. Each passenger is picked up by the assigned on-demand vehicle, driven to the desired location, dropped off, and finally released from the simulation. Each vehicle executes its plan until it becomes empty (i.e., all assigned passengers have been dropped off), and then it drives to park itself at the nearest station. In the simulation, any request that would be delivered to its destination with a greater delay than the maximum allowed delay $\delta_{\max}$ is considered a rejected request. However, as mentioned before, we configured the system so that these quality of service bounds are always satisfied, and consequently, there are no rejections during the simulation experiment.

Configuration and Scenarios used to Answer the Research Questions

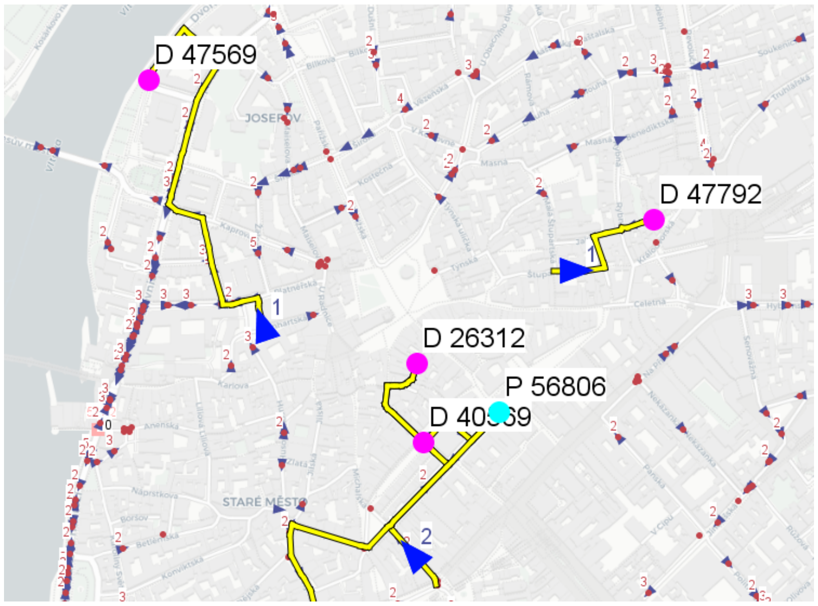
In our experiments, we considered the following six scenarios:

- *Present state*: All the requests are served by private vehicles. The vehicles are parked at the request's start location, i.e., there is no delay. The number of used vehicles is equal to the number of requests, and the total distance traveled is equal to the sum of the shortest paths between the origins of all requests and their destinations.

- *MoD w/o ridesharing*: MoD system without ridesharing, the plans are computed using



(a)



(b)

Figure 4.8: AgentPolis visualization of the simulated traffic in Prague during the traffic peak. Figure 4.8a (top): the entire city of Prague in the simulation. A more detailed (zoomed-in) view can be seen in Figure 4.8b (bottom). Vehicles are represented as blue triangles, with a number indicating the onboard passenger count. Red circles represent passengers. Some vehicles are highlighted, and their current plan is drawn with a yellow line. The pick-up and drop-off locations of the remaining actions are marked with cyan and pink circles, respectively, with a number indicating passenger ID. Note that in Figure 4.8b, there are some passengers already driving in two of the vehicles, so the number of drop-off locations is greater than the number of pick-up locations. The green triangles are vehicles that travel empty between stations (rebalancing).

IH, and the vehicle capacity is set to one, i.e., the passengers are not allowed to share rides.

- *MoD w. IH Ridesharing*: mod system with ridesharing computed by the IH.
- *MoD w. VGA Ridesharing (optimal)*: mod system with ridesharing computed by the VGA method to optimality.
- *MoD w. VGA Ridesharing (runtime limited)*: mod system with ridesharing computed by the VGA method, with the group generation time-limited to 60 ms per vehicle and the ILP solver maximum optimality gap of 0.5 %.
- *MoD w. VGA Ridesharing (PNAS)*: mod system with ridesharing computed by the vga method, with a set of timeouts/heuristics as described in Alonso-Mora et al. (2017), which we have reimplemented for this article.

In Manhattan, we do not evaluate the present state scenario, as the Manhattan dataset represents taxi trips, and therefore, the scenario with an MoD system without ridesharing is, in fact, also the "present state" scenario.

We simulate a morning peak time interval of 7:00-8:00 and an off-peak time interval of 11:00-12:00. To avoid the "cold start" artifacts, the simulation begins 30 minutes before the analyzed time interval, at 6:30 and 10:30, respectively, but for subsequent analysis, we only use the data captured after the thirty-minute start period. Including the 30 min warm-up time, there are 122 473 requests in the morning peak, and 42 633 requests in the off-peak experiment. The request-vehicle matching procedure is run every 30 s of the simulation. The maximum delay constraint $\delta_{\max}$ is set to 4 min, and the vehicle capacity is set to five passengers. The ILP solver in the VGA method computes the optimal solution with the maximum optimality gap of 0.02 %.

4.3 Results

We present the results here organized into six sections. We start with summary tables that contain the main results of the experiments, including the optimization criterion: total distance traveled by all vehicles (Section 4.3.1). Then, we discuss different aspects and quantities in detail, and we conclude with a performance analysis of the VGA method (Section 4.3.6). To run the experiments, we used a desktop system with Intel Core i7-8700K CPU (3.7 GHz, 6/12 physical/virtual cores) and 64 GB RAM.

4.3.1 Operating Cost and Computational Time

Tables 4.4 and 4.5 summarize the main results of the experiments. As explained in Section 4.2.2, we computed the size of the fleet to always guarantee full service availability. Since the service level is always 100 %, we do not show this metric in result tables and plots. The first row shows the value of our optimization criterion, i.e., the system operation cost measured in terms of total distance driven by the fleet vehicles. We can see that when using

	Present State	**Mobility-on-Demand**				
		No Ridesh.	IH	VGA	VGA lim	VGA PNAS
Optimal	-	-	no	yes	no	no
Total veh. dist. (km)	758 001	1 002 766	539 793	429 172	451 978	475 378
Avg. delay (s)	-	132	190	180	178	161
Avg. density (veh/km)	0.0077	0.0085	0.0053	0.0046	0.0048	0.0049
Congested seg.	8	25	1	1	1	0
Heavily loaded seg.	163	291	31	10	17	20
Used Vehicles	122 473	33 066	15 685	13 787	14 449	14 607
Avg. comp. time (ms)	-	181	18	192 903	27 714	15 598

Table 4.4: Main results from the considered scenarios during the morning peak (7:00-8:00). Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with density above 50 % of the critical density.

the VGA method instead of IH during the morning peak, we can save more than 110 000 km of vehicle distance driven, which represents more than 20 % reduction. Compared to the “no ridesharing“ scenario and to the present state, the VGA method saves over 573 000 km (57 %) and 328 000 km (43 %), respectively. Even in off-peak time, the VGA method can save about 17 % of the total distance driven compared to the IH, and about 48 % compared to the “no ridesharing” scenario.

The VGA method is considerably slower than IH. The average computational time per one optimization batch in the peak scenario was about 193 s, compared to 18 ms for the IH. Such a difference in the computational time may look extreme, but we have to consider the scale of the scenarios that were solved to optimality using the VGA method. The largest assignment problems (batches) contained more than 3000 waiting requests, 21 000 active requests (including passengers already driving to their destination), and 11 000 vehicles.

The runtime-limited experiment shows that we can speed up the VGA method significantly by merely limiting the computational time for the group generation and the solver. In the VGA limited experiment, we reduce the computation time more than six-fold over the unconstrained version of the VGA method while still reducing the total traveled distance by more than 16 % over the IH. The VGA PNAS experiment reduces the computational time by another 42 % at the cost of being closer to IH in the traveled distance (12 % improvement). In the off-peak scenario, the VGA limited performs almost the same as the unconstrained version because the time limits are rarely reached. Note, however, that there is more than a 5 % increase of traveled distance in VGA PNAS, despite similar computational times suggesting that this method is not suitable for scenarios where sufficient computational resources are available.

In Table 4.6, we can see the results of the Manhattan experiments. Our optimal implementation of the VGA method was able to compute the optimal assignments while the average computational time for a 30 seconds batch was less than 7 seconds. This is in con-

	Present State	Mobility-on-Demand				
		No Ridesh.	IH	VGA	VGA lim	VGA PNAS
Optimal	-	-	no	yes	no	no
Total veh. dist. (km)	283 483	344 613	211 285	175 865	176 957	186 520
Avg. delay (s)	-	131	191	179	179	165
Avg. density (veh/km)	0.0045	0.0047	0.0034	0.0032	0.0032	0.0032
Congested seg.	0	1	1	0	0	0
Heavily loaded seg.	5	10	3	1	2	2
Used Vehicles	42 633	7727	4646	4746	4802	5180
Avg. comp. time (ms)	-	4	1	5438	4408	4113

Table 4.5: Main results from the considered scenarios, off-peak (11:00-12:00). Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with density above 50 % of the critical density.

trast to results reported in Alonso-Mora et al. (2017) that were not computed to optimality and required more than 21 seconds to compute the most similar configuration ($q_{\max} =$ 5 minutes, vehicle capacity of four passengers, 3000 vehicles). This may be because the algorithm by Alonso-Mora et al. (2017) was developed and optimized to allow evaluation of scenarios with even larger delays of 7 minutes and with vehicle capacities of up to 10 passengers; such configurations result in an exponentially larger number of potential passenger-vehicle assignments and consequently, cannot be computed to optimality even with our performance-optimized VGA method.

Because the Manhattan experiment is less complex compared to the Prague experiment, we can observe a similar effect as in the Prague off-peak experiment: the VGA limited method computes only slightly worse solutions than the optimal method, and the computational times are similar as well. This is because the time limits of the VGA limited method were not reached in the majority of iterations.

Our re-implementation of the PNAS method gives a rather surprising result: the performance metrics are worse than the IH while using more computational time than the optimal method. We investigated this surprising result and found out that the cause is one of the heuristics that limits the number of vehicles considered for assignment to a particular request to the 30 nearest vehicles. This heuristic can limit the exploration so much that the solution can be worse than the IH solution. Moreover, for less complex scenarios, the time needed to compute the 30 nearest vehicles can dominate the total computational time, as it happened in our case, probably because this heuristic was not optimized. This observation suggests that in order to achieve acceptable performance, one may need to vary the parameters of heuristics based on the complexity of the problem instance at hand. We also performed another experiment using the VGA PNAS method with this heuristic turned off. For results, see numbers in parentheses in the last column of result tables (4.6, 4.7).

Table 4.7 shows another set of results of experiments with $q_{\max} = 7$ minutes and the

	No Ridesh.	**IH**	**VGA**	**VGA lim**	**VGA PNAS***
Optimal	-	no	yes	no	no
Total veh. dist. (km)	868 899	362 387	334 195	334 737	377 563 (344 057)
Avg. delay (s)	109	117	109	109	83 (110)
Avg. density (veh/km)	0.0183	0.0085	0.008	0.008	0.0089 (0.0082)
Congested seg.	220	9	9	8	14 (10)
Heavily loaded seg.	692	129	117	112	152 (115)
Used Vehicles	46 186	20 272	19 714	19 712	22 545 (20 293)
Avg. comp. time (ms)	918	57	6646	6650	24 717 (7170)

Table 4.6: Main results from the Manhattan scenarios during the peak (19:00-20:00) with a maximum passenger delay of 4 minutes. Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with a density above 50 % of the critical density. For the VGA PNAS method, we also tested a version that does not limit the vehicles considered for each request to 30 nearest vehicles (in parentheses).

capacity of 10 persons per vehicle, which corresponds to the most complex configuration in Alonso-Mora et al. (2017). In this experiment set, we only evaluated the three sub-optimal ridesharing methods to see how they behave under such parametrization. Interestingly, for this scenario, the IH achieves the best performance: The IH finds plans with a total travel distance that is 12 % smaller than plans found by both sub-optimal versions of the VGA method using only a fraction of computational resources. This experiment demonstrates the limit of applicability of the VGA method for routing in large-scale MoD systems. The relaxed time windows and increased vehicle capacity increase the number of feasible groups and the maximum group size to a level that cannot be solved by the ILP solver and the single-vehicle solver, respectively, in practical time. Consequently, the VGA algorithm is unable to return an optimal solution to such instances.

4.3.2 Trade-off Between Operating Cost and Passenger Discomfort

Another metric that we tracked is the service quality, represented by the passenger delay relative to transportation by the private vehicle. From Tables 4.4 and 4.5, we can see that the optimal VGA method saves about 5 % time over the IH in both peak and off-peak experiments. The trade-off between the operating cost (distance traveled) and the service quality (average delay) is depicted in Figure 4.9.

A more detailed overview of the passenger delays with a delay histogram for the four MoD scenarios in both time windows is in Figure 4.10. It is clear that for both peak and off-peak, the VGA method reduces the passenger delay resulting from ridesharing compared to the IH. Nevertheless, even in the case of the VGA method, there is a noticeably greater

	IH	**VGA lim**	**VGA PNAS***
Optimal	no	no	no
Total veh. dist. (km)	233 859	275 028	267 471
Avg. delay (s)	227	224	217
Avg. density (veh/km)	0.006	0.0067	0.0066
Congested seg.	0	1	0
Heavily loaded seg.	24	53	48
Used Vehicles	13 319	16 517	16 025
Avg. comp. time (ms)	25	57 554	139 811

Table 4.7: Main results from the Manhattan scenarios during the peak (19:00-20:00) with a maximum passenger delay of 7 minutes and vehicle capacity of 10 persons per vehicle. Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with a density above 50 % of the critical density. For the VGA PNAS method, we used the version that does *not* limit the vehicles considered for each request to 30 nearest vehicles.

delay compared to the no ridesharing scenario, where the delay can occur only before the passenger is picked up or over the present state, where there is no delay because a car is assumed to be available at the origin of each passenger trip.

4.3.3 Impact of MoD on Congestion

In addition to the operational cost, we analyzed the impact of the MoD system on congestion. We consider road segments with traffic density above the critical density of 0.08 vehicle m^{-1} (Tadaki et al., 2015) as *congested*. Segments with density above 0.04 vehicle m^{-1} are considered as *heavily loaded*. As you can see in Table 4.4, in the morning peak, using the optimal VGA method reduces the average traffic density by 13 % over the ridesharing that uses IH, and by 46 % and 40 % over the MoD without ridesharing and the current state, respectively. We can see the same trend when we look at the number of congested and heavily loaded segments. In the off-peak experiment, the situation is similar, but the absolute numbers indeed show that there is no congestion in any of the scenarios. Finally, Figures 4.11 and 4.12 depict traffic densities on every road for all five scenarios.

4.3.4 Fleet Size and Vehicle Occupancy

Also, for each scenario, we recorded the number of vehicles that were used at least once during the simulation. For the present state scenario, we consider a dedicated vehicle for each request. Therefore, the number of used vehicles is equal to the number of requests. The results confirm that the VGA method indeed makes the mod system more efficient. During the peak hour, the optimal VGA used 1898 (12 %) fewer vehicles than the IH. Compared to

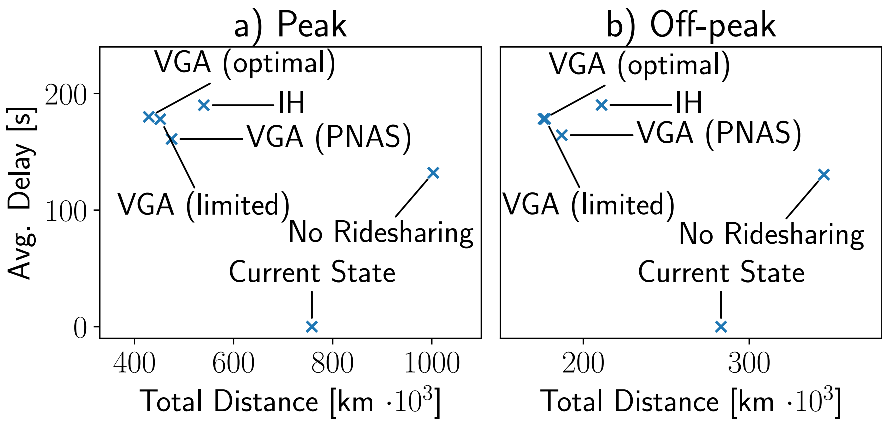


Figure 4.9: The trade-off between the total distance traveled by all vehicles and the average delay of one passenger trip.

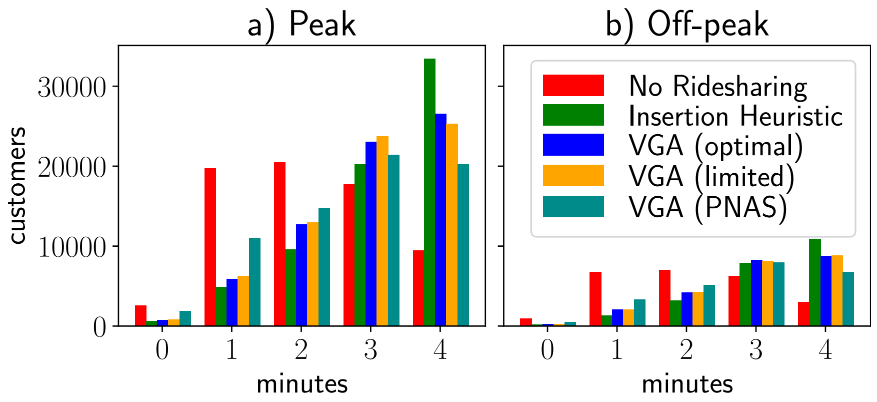


Figure 4.10: Histograms of delays. The present state scenario is omitted as the delay is always zero.

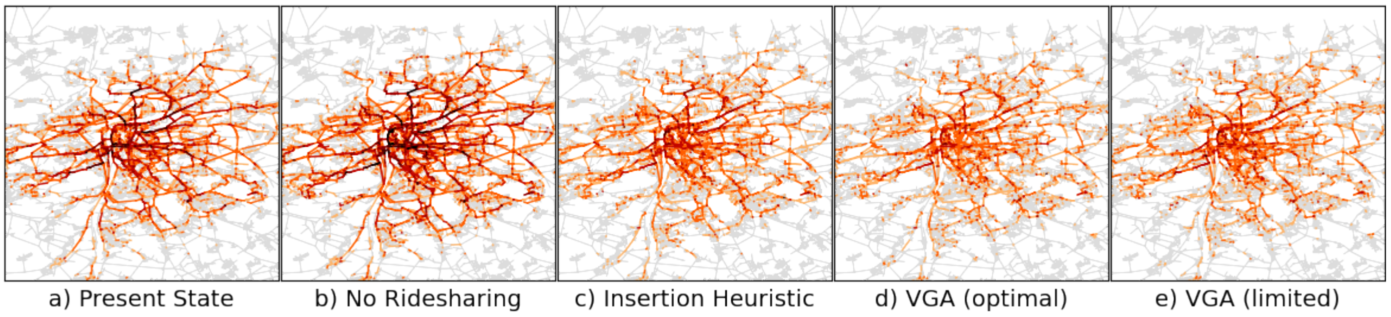


Figure 4.11: Traffic density map of the four scenarios during the morning peak. Darker colors signalize higher traffic density. The black color means that the road segment is congested. We omit the density map for the VGA PNAS experiment from this figure as it is very similar to the density map for the VGA limited experiment.

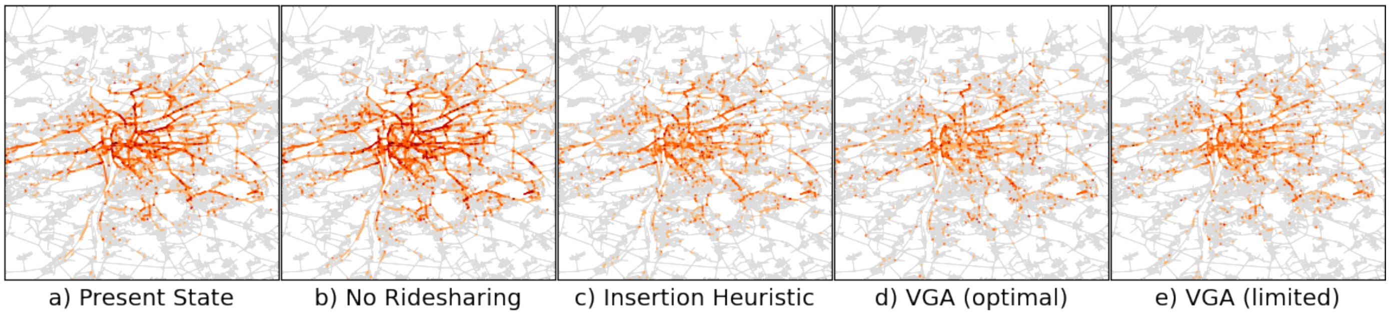


Figure 4.12: Traffic density map of the four scenarios during the off-peak time. Darker colors signalize higher traffic density. The black color means that the road segment is congested. We omit the density map for the VGA PNAS experiment from this figure as it is very similar to the density map for the VGA limited experiment.

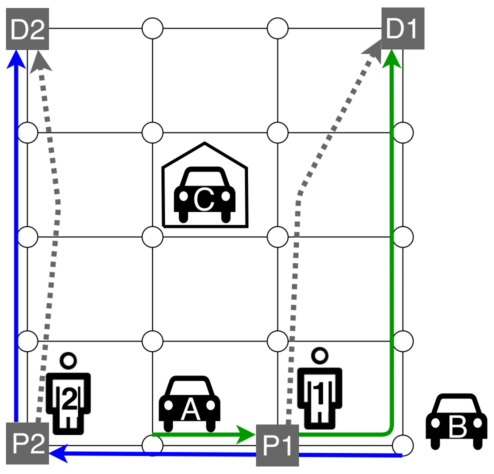
the MoD system without ridesharing, the MoD system with optimal ridesharing used about one-third of the vehicle fleet, and compared to the present state system, the reduction is almost thirteen-fold.

In the off-peak time, however, we registered that the optimal VGA method uses about 2 % more vehicles than IH. By analyzing the simulation output, we found an explanation for this perhaps surprising result. First, counterintuitively, it is possible that a suboptimal vehicle assignment (generating plans with longer total distance) can lead to fewer vehicles being used, as illustrated in Figure 4.13. Second, by analyzing the vehicle trips in both IH and VGA scenarios, we found that such situations occur frequently due to unbalanced demand. In other words, the optimal method uses more vehicles not despite, but because its plans are more operating cost-efficient: the vehicles simply serve requests too quickly, which increases the chance of ending up in the areas with lower demand, where they need to wait a long time before another request appears nearby. This reminds us that to fully understand MoD systems, we need to study not only operation cost vs. service quality trade-offs but also operation-cost vs. capital cost trade-offs associated with different design and control strategies.

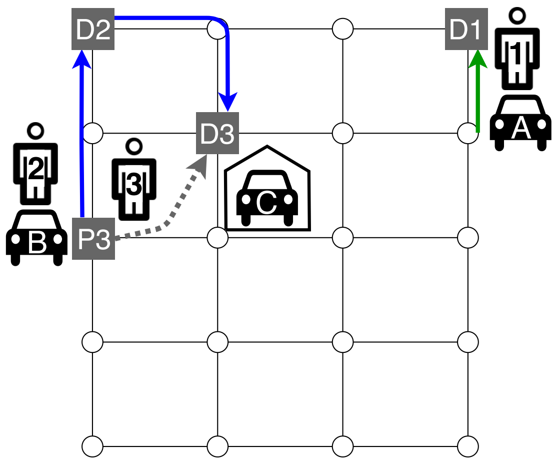
Next, we measured vehicle occupancy: Figure 4.14 shows the occupancy histogram for the four compared scenarios. We can see that vehicle occupancy is the highest when using the optimal method in both peak and off-peak scenarios.

4.3.5 Sensitivity Analysis of the VGA Method

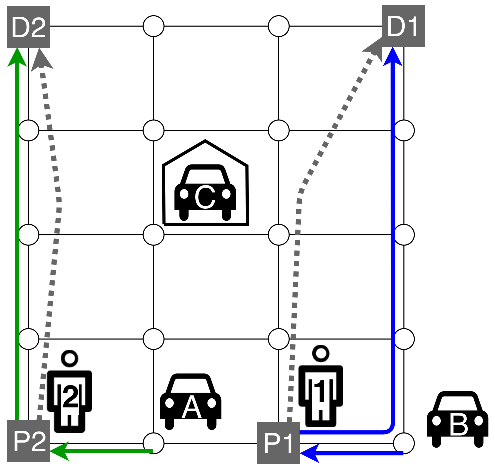
We analyze the sensitivity of the variation in batch length, maximum delay, and capacity with respect to total traveled distance, computation time, and average passenger delay. Note that the time between a request announcement and the end of the batch, when the passenger-vehicle assignment is recomputed, counts towards the delay of the request. Therefore, for scenarios with longer batches, we also extended the maximum delay in order to keep the average effective maximum delay of 4 minutes. Also, note that we use the same stations and fleet for all experiments, and consequently, some requests were rejected in configurations



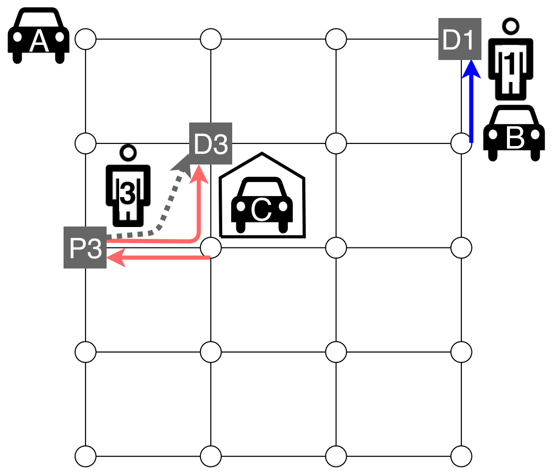
(a) IH Iteration 1



(b) IH Iteration 2



(c) VGA Iteration 1



(d) VGA Iteration 2

Figure 4.13: Example of the capital cost paradox. In Figures 4.13a and 4.13b, we can see two iterations of the IH. In Figure 4.13a, there are three vehicles: vehicles A and B, and vehicle C that resides in the station, representing a potentially unlimited pool of vehicles. Also, there are two passengers (1 and 2), that request travel from their current locations $P1$ and $P2$ to their destinations $D1$ and $D2$ (denoted by dashed arrows). Solid arrows denote the plans for both vehicles computed by the first iteration of the IH. In Figure 4.13b, there is the same scenario in the next iteration. Both cars moved by five steps in the grid, and also, a new request appeared. We can see the new plans generated by the second iteration of IH too. The second set of Figures ((4.13c) and (4.13d)) shows the exact same two iterations solved by the VGA method. Note that although we saved one segment of traveled distance (vehicles traveled 14 segments in the grid combined compared to 15 segments in the case of the IH), we used one extra vehicle (vehicle C) that was not needed in the IH scenario, thus effectively increased the required fleet.

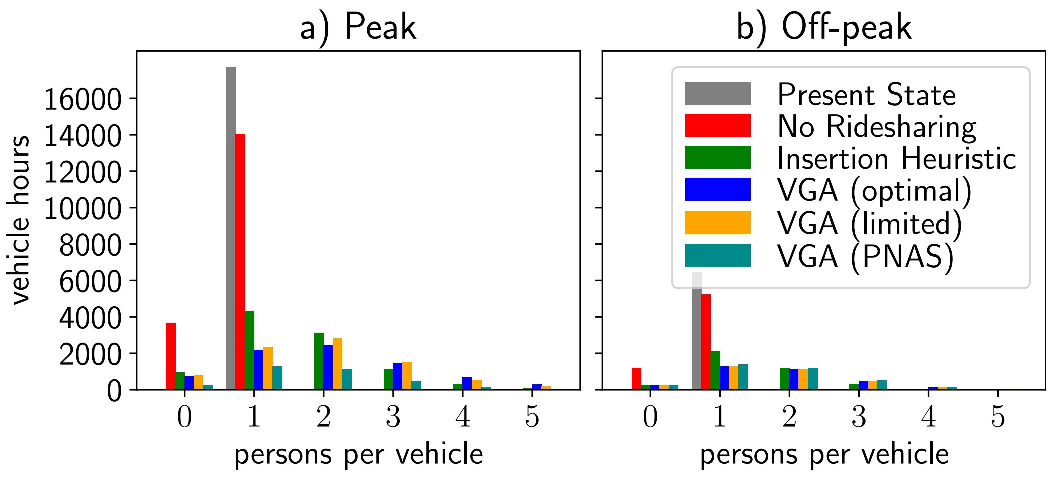


Figure 4.14: Occupancy histogram of all five scenarios.

with shorter maximum delays or longer batch lengths. However, the service level remains above 99 % in all configurations, so the impact of rejected requests on the results is negligible.

We can see the results in Figure 4.15 (peak scenario) and Figure 4.16 (off-peak scenario). In the peak scenario, we were able to compute the optimal solution only for batch length of up to 30 seconds and for up maximum delay of up to 4 minutes. For larger values, the algorithm failed to terminate within 24 hours. As expected, the runtime of the optimal method grows exponentially with maximum delay. This is best seen in the case of the off-peak experiment, where the algorithm was able to find an optimal solution within 5 minutes of runtime on average for the 6-minute maximum delay but failed to compute optimal solutions within 24 hours of runtime for the 7-minute maximum delay. Clearly, the optimal method would not scale to scenarios with larger limits on maximum delay, and one of the resource-limited variants would have to be employed.

We can observe that the system's efficiency (measured in terms of total distance driven) monotonically increases with maximum allowed delay for all considered methods with the exception of the VGA limited method, which achieves low runtimes by prematurely terminating computation in several stages of the algorithm. As we can see, the values of cut-off parameters that work well for a 3-5 minute maximum delay lead to inferior performance for larger maximum delays. We can also observe that with the increasing maximum delay, the gap between the optimal method and both resource-constrained VGA methods increases. Remarkably, our experiments show that for high values of max delay, the IH achieves almost identical performance as the VGA PNAS algorithm while using only a fraction of computational resources.

The batch length negatively impacts the average travel delay experienced by passengers because they need to wait until the end of the batch for their request to be assigned to a vehicle. The motivation for using longer batch lengths is to gather more requests and to find a more efficient passenger-vehicle assignment. However, it appears that even in the off-peak experiment, these efficiency gains are only realized using the optimal solution method. For suboptimal solution methods, the efficiency gains are either negligible or straight-out negative. The higher vehicle capacity also increases the potential for ridesharing, which, in turn, could improve the operational efficiency of the system. In the peak experiment, the increased complexity prevented the computation of the optimal solution for vehicle capacity set to 10 passengers. However, our results show that the resource-limited variants were not able to improve the solution significantly. This can be caused by reaching the time limits before the algorithm can generate high-occupancy plans, or it can indicate that the capacity of 5 passengers per vehicle is sufficient. For reference, in the off-peak experiment, we were able to compute optimal solutions, and our results show that the vehicle capacity of 5 is sufficient for the off-peak demand intensity. However, it is still possible that high-capacity vehicles would be better utilized when planning for peak-intensity demand.

Concerning the suboptimal versions of the VGA method, our VGA limited method computes higher quality solutions for shorter batch lengths and shorter maximum travel delays, while the PNAS version of the VGA method achieves better performance for longer batch lengths and longer maximum delays. This is probably because the PNAS version uses IH to compute plans for groups larger than four (see the supplemental material of Alonso-Mora et al. (2017)). This will negatively affect solution quality for easier instances. However, for harder instances, this approach may be beneficial compared to our strategy because it allows forming larger groups with potentially suboptimal plans. This observation suggests that a resource-constrained VGA method should use a heuristic to compute larger groups, but the threshold for using this heuristic should be determined by the remaining computational time.

Finally, we performed a sensitivity analysis for the Manhattan case study: the results are reported in Figure 4.17. It tells a similar story as the sensitivity analysis for the Prague case study. Some of the previously discussed phenomena are even more apparent in the Manhattan sensitivity analysis. We can see that the computational requirements grow with the maximum delay not only for the optimal VGA method but also for the resource-constrained VGA methods. Also, we can clearly see that the efficiency (total distance driven) gap between IH and the constrained VGA methods is shrinking for larger maximum delays. For the maximum delay of 7 minutes, the IH method starts to outperform both resource-constrained VGA methods.

4.3.6 Computational Time Analysis of the VGA Method

Finally, we inspect the computational requirements of the VGA method. In Figure 4.18, we show the evolution of the number of active requests (top), maximum computed group size (middle), and computational time for group generation and group-vehicle assignment process (bottom) during the peak scenarios, including the warm-up period. Looking at the maximum group size, we see that the 60 ms limit for the group generation results in groups of the maximum size of 5-7 in most batches, while in the optimal scenario, the maximum group size has high variance and goes up to 11.

When we compare the maximum group size with the computation times, we can obtain other valuable insights: a) the group generation time is strongly dependent on the maximum group size, and thus it has low variance in the limited scenario and high variation in the optimal scenario, b) the solver time does not depend on maximum group generation time

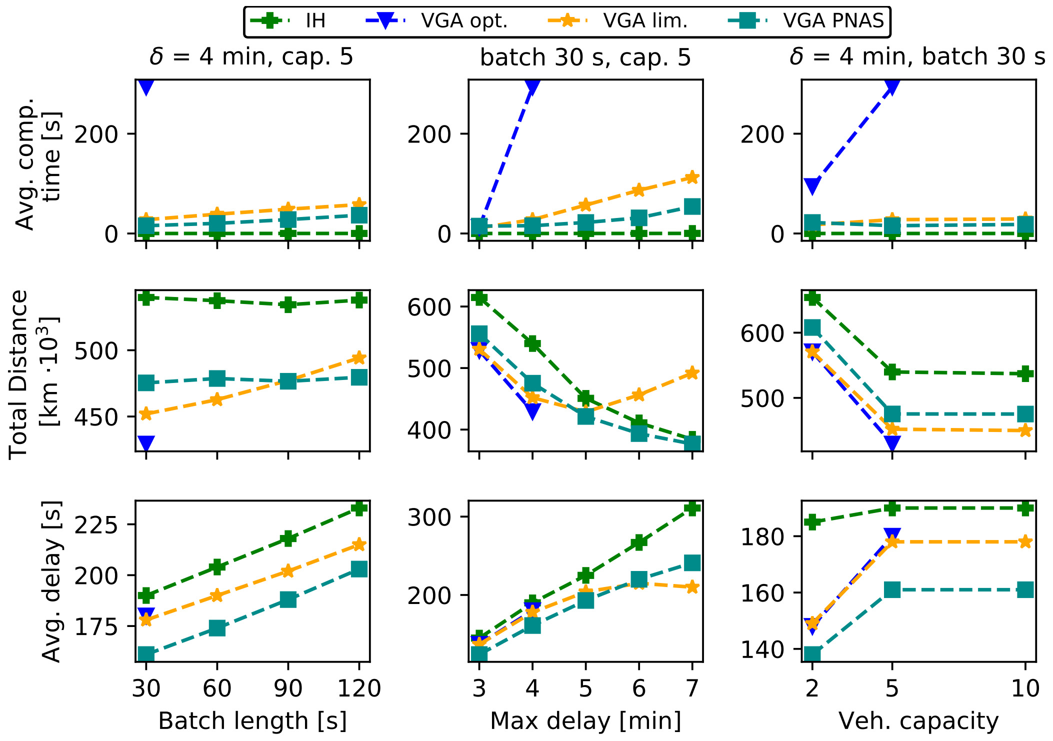


Figure 4.15: Sensitivity analysis: peak. Each column represents one experiment set, and inside each column, each value on the x-axis represents one experiment. Each row displays a single measured quantity. The optimal VGA method is only computed for batch length 30 s, maximum delay of 3 and 4 minutes, and capacity of 2 and 5 persons per vehicle. For other parameter values, the optimal VGA method did not terminate within 24 hours.

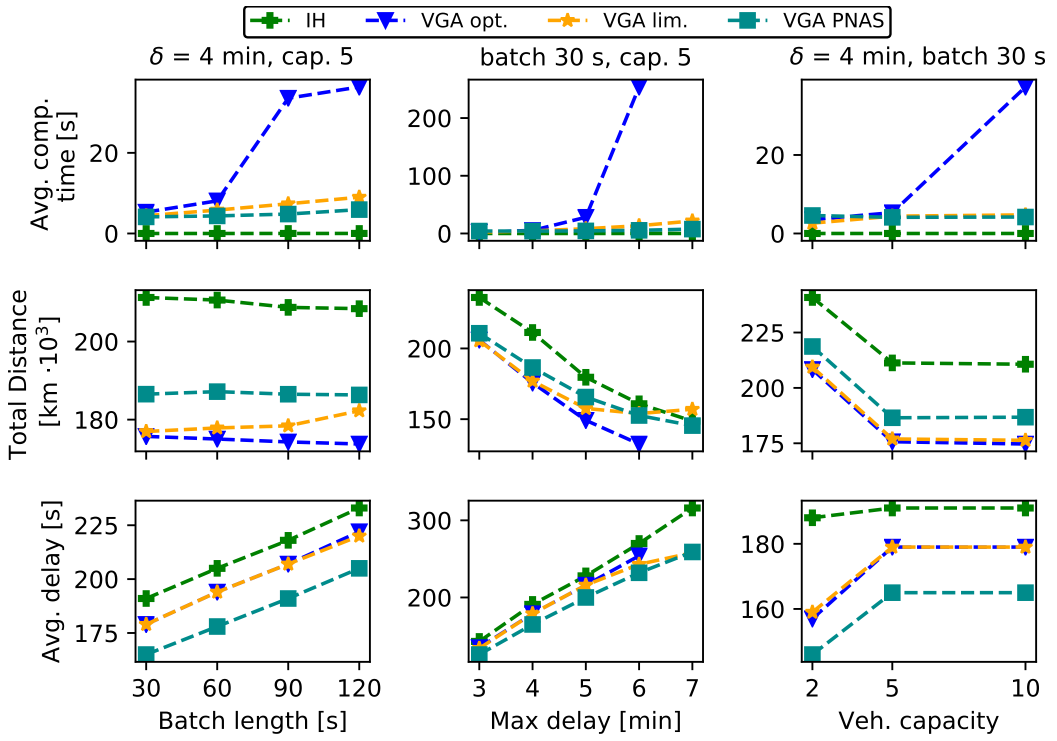


Figure 4.16: Sensitivity analysis: off-peak. Each column represents one experiment set, and inside each column, each value on the x-axis represents one experiment. Each row displays a single measured quantity. The optimal VGA method is only computed for the maximum delay of up to 6 minutes. For the maximum delay of 7 minutes, the optimal ridesharing assignment cannot be computed within the 24-hour limit.

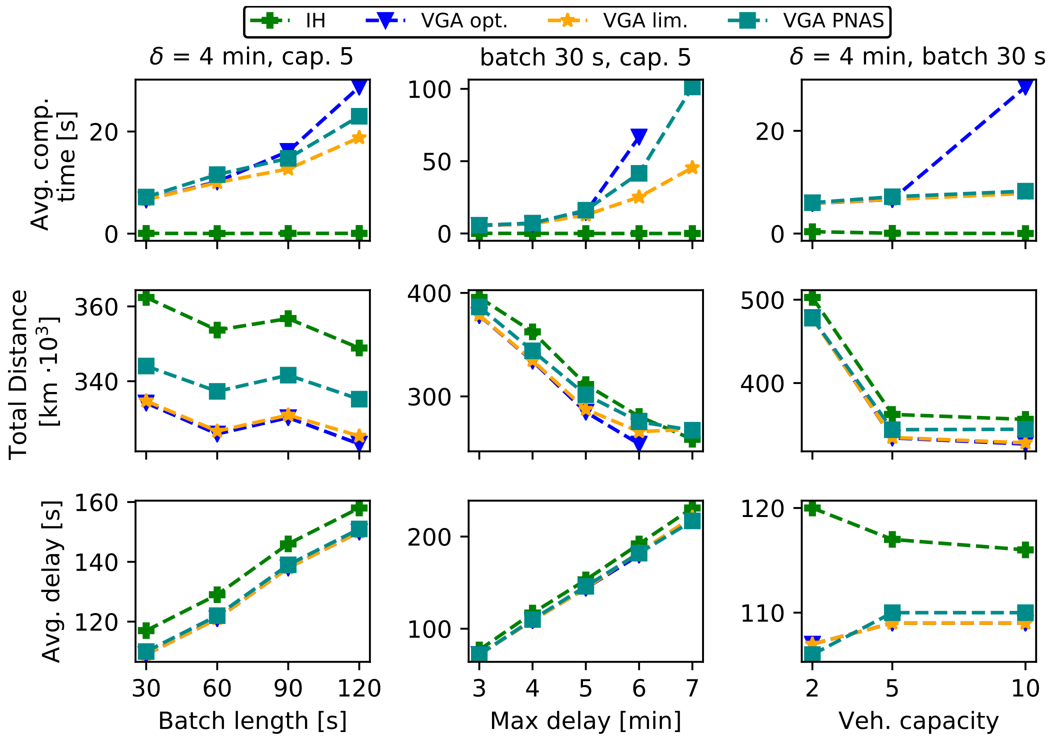


Figure 4.17: Sensitivity analysis: Manhattan. Each column represents one experiment set, and inside each column, each value on the x-axis represents one experiment. Each row displays a single measured quantity. The optimal VGA method is only computed for the maximum delay of up to 6 minutes. For the maximum delay of 7 minutes, the optimal ridesharing assignment cannot be computed within the 24-hour run time limit.

much, and it is highly variable in both limited and optimal variant, and c) the group generation time dominates in both scenarios. These findings suggest that further performance optimization of the group generation process may lead to a more favorable trade-off between the solution cost and computational time.

4.4 Summary

Mobility-on-demand systems represent a promising alternative to private car transport. Already, there are many such systems in operation, mostly in the form of ride-hailing services provided by Transportation network companies like Uber or Bolt. As more and more people use these services and more operators are emerging, it becomes increasingly important to understand the impact of these systems and likewise to understand how to operate them efficiently. In this chapter, we try to answer three research questions related to the operation of large-scale MoD systems. To answer these questions, we use a detailed multi-agent simulation of a hypothetical large-scale MoD system that replaces all private cars in a city.

The first question is about the impact of the large-scale deployment of MoD systems. This question is relevant even to the current MoD systems as there are areas where the MoD systems already account for a significant part of the total transportation. The results confirm that the MoD systems can reduce the number of vehicles needed to serve the existing travel demand. The reduction is 4 to 6-fold, depending on the scenario. However, even the risks of increased traffic intensity are confirmed: the total distance traveled by vehicles increases by 20 % to 30 % compared to the current system with private cars. This extra distance is caused by both empty trips between passengers and rebalancing trips. These extra trips also increase traffic congestion, as we can see in the traffic density maps.

The second question asks how much ridesharing can improve the performance of large-scale MoD systems. Current MoD systems are already experimenting with ridesharing, but answering this question can help to understand its potential and accelerate its deployment. As expected, the reduction in the total distance traveled is significant: the MoD system with simple heuristic ridesharing reduces the total distance traveled by 46 % compared to the MoD system without ridesharing, and by 22 % compared to the current system with private cars. Moreover, there is an additional reduction in the number of vehicles needed to serve the travel demand: in general, about half of the vehicles can be removed from the fleet.

Finally, we asked how much we can further improve the performance of the MoD system by computing the ridesharing assignments optimally. Most of the time, heuristic methods are used to compute the ridesharing assignments. This makes sense because the problem is fundamentally complex and consequently optimal methods are typically too slow. However, knowing the optimal solutions has a benefit at least from the theoretical point of view: it can help us to understand the potential of the improvement for the heuristic methods. When answering the third research question, we were able to deliver an optimized implementation of the Vehicle-group Assignment method that can compute even large instances to optimality. The result shows that there is a significant gap between the optimal and heuristic solutions: the optimal ridesharing assignments reduce the total distance traveled by about

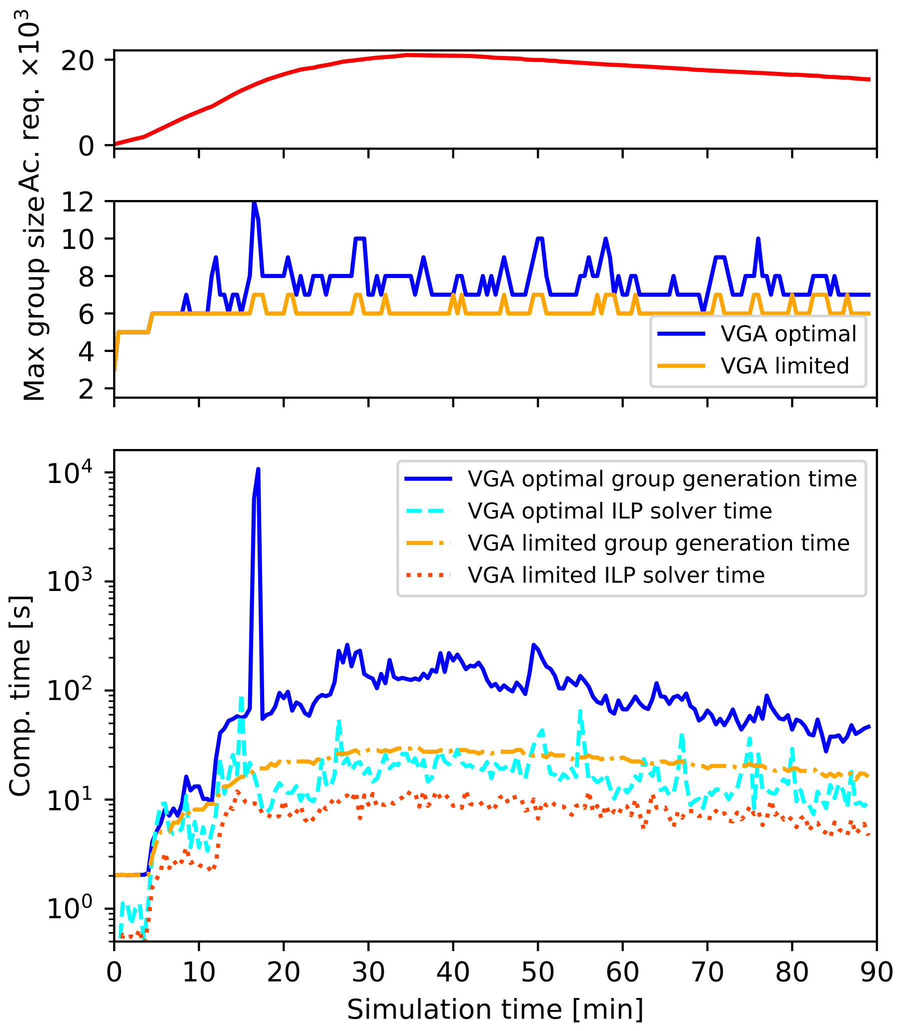


Figure 4.18: Computational efficiency analysis of the VGA scenarios during the peak time. In the top figure, we show the evolution of the number of active requests over time. We can see that after the warm-up time, the number of active requests in the system is stable, only slowly decreasing. The middle figure displays the maximum group size that was computed in each batch. The bottom figure demonstrates how the computational time, consisting of the group generation time and the ILP solver time, changes during the simulation.

20 % compared to the Insertion Heuristic assignments, while simultaneously reducing the average passenger delay by 5 %. To test some method that represents a trade-off between the optimal method and IH, we developed a resource-constrained VGA method. This method provides more than 16 % travel distance saving over IH while reducing the computational time by almost 90 % compared to the optimal solution. Moreover, we achieved further important insights when applying sensitivity analysis.

To summarize the findings of this chapter, we can say that the MoD systems, while having the clear benefit of reducing the number of vehicles needed, also increase the total amount of traffic in the system due to empty trips. This can be, however, mitigated and even reversed by implementing large-scale ridesharing. Finally, there is a significant gap between heuristic and optimal ridesharing methods, so it makes sense to invest in improving the dispatching methods. Specifically, the future heuristic methods should address the scenarios for which the optimal method is slow: the scenarios with a long time horizon and wide time windows.

Another important finding of this thesis is that even for very large transportation scenarios, a simulation is a viable tool to answer the research questions. Even with tens of thousands of vehicles and requests at the same time, the simulation can be run on an ordinary computer much faster than in real-time. Additionally, this suggests that a higher fidelity of the simulation achieved by using a more detailed traffic model may be viable. Therefore the simulation can and should be used to answer future MoD-related research questions.

CHAPTER 5

Optimal Plan Chaining for Mobility-on-Demand

Our last goal was to use the knowledge from our simulation studies to develop a new heuristic method for vehicle dispatching in large-scale MoD systems. Specifically, we aimed to develop a method that would be able to solve DARP instances that cannot be solved optimally in a reasonable time, as the solutions of the resource-constrained methods and other evaluated heuristics are significantly worse than the optimal solutions (as we showed in the last chapter).

We were researching the possible directions based on both literature and analysis of the characteristics of problematic DARP instances. We identified that one of the deciding factors for the complexity of the DARP is the long time horizon of the instance. Based on that, we developed a new heuristic method that is based on the principle of 1) splitting the problem into subproblems with a shorter time horizon, 2) solving the subproblems optimally, and 3) chaining the resulting plans to obtain the final solution. While the first step is straightforward, and there are existing optimal methods for the second step, we have not found any optimal method for the plan chaining problem. In fact, we have not found even a plan chaining formulation that would consider the time windows that are crucial for most vehicle routing problems. Therefore, we decided to formulate the plan chaining problem with time windows and develop a method that would solve it optimally.

In this chapter, we present both our research on the plan chaining problem and also the new DARP heuristic method that uses it as a subproblem. First, we introduce the plan chaining problem and show that it appears as a subproblem of many MoD-related problems (5.1). Next, we formulate the problem in Section 5.2. Then, we present our method for

This chapter is based on the manuscript titled "Optimal Chaining of Vehicle Plans with Time Windows" submitted to the journal *Transportation Research Part C: Emerging Technologies*. Preprint:

Fiedler, D., Difonzo, F. V., & Mrkos, J. (2024, February 24). *Optimal Chaining of Vehicle Plans with Time Windows*. arXiv: 2401.02873 `[cs, math]`. https://doi.org/10.48550/arXiv.2401.02873

For all author's publications together with the citation counts and research contributions, see Appendix A.

solving the plan chaining optimally (Section 5.3). Finally, we present the DARP heuristic method that uses the plan chaining as a subproblem, and demonstrate its capabilities on case studies of four areas: New York City, Manhattan, Chicago, and Washington, DC (Section 5.4).

5.1 Research on the Plan Chaining

One of the key problems regarding MoD systems is to determine the minimal vehicle fleet able to serve all travel requests: the *fleet sizing* problem. In the well-known 2018 article, Vazifeh et al. (2018) solves the problem optimally using two steps. First, they generate a shareability network: a graph where the nodes are the plans for solving each request, and edges are the possible connections between those plans. The second step is to minimize the number of edges in the shareability network. The authors prove that the shareability network is an acyclic graph, and thus, the problem can be solved in polynomial time by the Hoptcroft-Carp algorithm applied to a bipartite graph corresponding to the shareability network.

Inspired by this work, Wang et al. (2021) propose a method to solve the fleet-sizing problem in a ridesharing context. The authors formalize the problem as finding a minimum cover in the set of all possible dispatching graphs: the dispatching tree cover. They prove that the dispatching tree cover problem is NP-Hard and propose a heuristic algorithm; then, they evaluate its performance both analytically and on a case study in Shenzhen.

Xu et al. (2022) extend Vazifeh et al. (2018) by considering the limitation of the number of available vehicles in particular zones. To achieve this, they transform the shareability network into a min-cost flow problem, which also incorporates vehicles as network nodes. The zone constraint is represented by edges between vehicles and requests: these exist only if the request originates in the same zone where the vehicle is present. As the authors point out, the formulated min-cost flow problem can be solved in polynomial time by the network simplex algorithm.

Similarly to Wang et al. (2021), Qu et al. (2022) also propose to incorporate ridesharing into the fleet-sizing problem. Contrary to their predecessors, however, they decouple the problem: first, they solve the ridesharing (DARP) problem, and next, they apply the method from Vazifeh et al. (2018) on the graph composed from the computed ridesharing plans. Additionally, the matching phase computes the utility with regard to an anticipated travel demand computed from an ensemble model. Decoupling the problem breaks the guarantee of optimality. However, it means that the minimal path cover in the shareability graph can be solved in polynomial time, as in Vazifeh et al. (2018).

Apart from fleet-sizing, shareability networks can be used to solve other problems related to the MoD. The concept of the shareability network itself has been introduced in Santi et al. (2014). The authors use a sharability network to study the pairwise sharability trips in Manhattan. Note that, differently than in the fleet sizing articles, here the trips are connected in a sharability network not only in case they can be connected consecutively, but also if they can share the ride. In conclusion, we obtain an optimal solution for the ridesharing (DARP) problem limited to two persons per vehicle, provided that we optimize the shareability network using the minimum-weighted matching, as the article proposes.

Alonso-Mora et al. (2017) then lift the two-passenger limit and propose an optimal algorithm to solve the online ridesharing (DARP) problem. However, due to the computational complexity, they introduce several heuristic relaxations to compute the solution in a reasonable time. Later, Čáp and Alonso-Mora have analyzed the trade-off between the operation cost and service quality with this method, which they have called the vehicle-group assignment method (VGA) Čáp and Alonso-Mora (2018).

Finally, in Chapter 4, we examine the optimal version of the VGA algorithm from the perspectives of system efficiency and computational time. An important finding is that the optimal assignments can be computed in practice only if we limit the time horizon of the instance, i.e., if we try to match only requests with origin times within a short time. At the same time, we show that the relaxed version performs poorly if the time horizon is longer, being outperformed by the baseline method: the insertion heuristic. This leads us to a question: could we obtain a better solution by computing the matching over a short time horizon and then connecting the resulting plans by solving the min-weight matching over the shareability network?

5.2 The Plan Chaining Problem

Here, we formulate the problem of *chaining* plans with time windows. The formulation is independent of the "real-world" problem, i.e., we can use the same formulation for fleet-sizing, dispatching, and potentially other problems with a similar structure. The main difference between our formulation and the previous formulations (Qu et al. (2022) and Vazifeh et al. (2018)) is that we allow plan delays that respect the given time windows. The previous formulations chain plans with fixed plan start/end times, despite the underlying problem being some variant of vehicle routing problem with time windows. Because the time-windows extension makes the network formulation quite complex, we do not use the network formulation, in contrast to previous works (Qu et al., 2022; Vazifeh et al., 2018; Xu et al., 2022). Instead, in this section, we propose a more compact formulation (see Problem 5.1 later in this section). However, the network formulation, loosely connected to the shareability network introduced in Vazifeh et al. (2018), is a backbone of the proposed method, and it is presented in Section 5.3.

We say that the chaining problem is a problem of transforming given vehicles and plans to the minimum cost chains, that consist of one vehicle and one or more plans, such that all the time constraints are met, each vehicle is at the beginning of no more than one chain, and all plans are part of exactly one chain.

For the purpose of plan chaining, the internal structure of the plan (pickup and drop-off times and locations) is irrelevant. Instead, each plan p from the set of plans P is defined by the origin time t_p^{o}, destination time t_p^{d}, and maximum delay $\delta_p^{\max}$ which represent the plan's time window. Each vehicle v from the set of vehicles as V is determined by its start time t_v^{s}.

We assume that travel times between all plans and vehicles a and b are known in the form of a travel time function $f_{\mathrm{tt}}(a,\ b), a, b \in P \cup V$. Note that the function parameter order matters here, as the travel time is often asymmetrical in real use cases. Analogously, we define the travel cost function $f_{\mathrm{tc}}(a,\ b)$, which represents the travel cost (if we optimize for minimum travel time, we can set $f_{tc} = f_{tt}$).

To use the time flexibility offered by the plans' maximum delay, we introduce *delayed plan variants*. For each plan p, there is a set of possible delayed plan variants Φ_p. We denote the union of all such sets as Φ. Each p' in Φ_p is defined by the delay $\delta_{p'} \leq \delta_p^{\max}$ over the original plan. As consequence it holds that $t^{\mathrm{o}}_{p'} = t^{\mathrm{o}}_p + \delta_{p'}$ and $t^{\mathrm{d}}_{p'} = t^{\mathrm{d}}_p + \delta_{p'}$.

We use the term *chaining* because we create sequences called *plan chains*.

Definition 5.1. A plan chain $c = \{c_i\}_{i \in \mathbb{N}}$ is a sequence such that:

$$c_1 \in V \tag{5.1}$$

$$\forall c_i \in c \setminus \{c_1\}, \quad c_i \in P \cup \Phi \tag{5.2}$$

$$t^{\mathrm{s}}_{c_1} + f_{\mathrm{tt}}(c_1,\ c_2) \leq t^{\mathrm{o}}_{c_2} \tag{5.3}$$

$$\forall i \in [3, |c|], \quad t^{\mathrm{d}}_{c_{i-1}} + f_{\mathrm{tt}}(c_{i-1},\ c_i) \leq t^{\mathrm{o}}_{c_i} \tag{5.4}$$

A set of all plan chains that can be constructed from vehicles V plans P (and associated delayed plans Φ generated from P) will be denoted as $\mathcal{C}(P, V)$.

Finally, we define the *chaining problem*:

Problem 5.1 (Plan Chaining)**.** Given a set of plans P and a set of vehicles V, compute the set of plan chains C for which the travel cost between consecutive elements is minimal:

$$\min_{C \subset \mathcal{C}_P^V} \sum_{c \in C} \sum_{i=2}^{|c|} f_{\mathrm{tc}}(c_{i-1},\ c_i) \tag{5.5}$$

such that exactly one variant per each plan is contained in exactly one chain and each vehicle is at the beginning of no more than one chain:

$$\sum_{c \in C} \sum_{i=2}^{|c|} \mathbf{1}_{\Phi_p}(c_i) = 1, \quad \forall p \in P \tag{5.6}$$

$$\sum_{c \in C} [c_1 = v] \leq 1, \qquad \forall v \in V \tag{5.7}$$

5.2.1 Chaining Problem Configuration for Solving Various Transportation Problems

We can solve various problems related to on-demand systems by solving the chaining problem with different configurations.

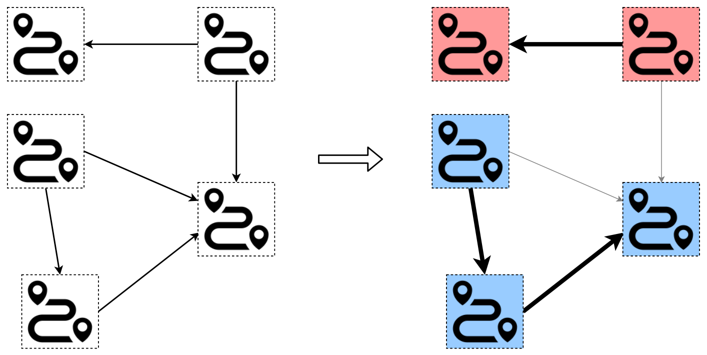


Figure 5.1: Example of using the minimum path cover to solve the fleet-sizing problem. On the left, there is a vehicle shareability graph. An arrow between any two plans signals that these plans can be served sequentially. On the right, there is the minimum path cover. Each color represents a chain of plans to be served by one vehicle. The connections (arrows) between plans in the chain are bold.

For fleet-sizing, we create a dedicated vehicle for each plan and set the travel cost to a constant for every used vehicle:

$$f_{\mathrm{tc}}(a,\ b) = \begin{cases} 1, & \text{if } a \in V, \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

This way, the number of vehicles will be minimized without considering the travel cost between plans.

For minimum cost dispatching (DARP), we just set the $f_{\mathrm{tc}}(a,\ b)$ to capture the cost of travel between a and b; for example, we can set:

$$f_{\mathrm{tc}}(a,\ b) = f_{\mathrm{tt}}(a,\ b) \quad \forall a, b \in V \cup P \cup \Phi. \tag{5.9}$$

To prevent long waiting times, we can set $f_{\mathrm{tc}}(a,\ b) = \infty$ for all a and b where the $t^{\mathrm{o}}_b - t^{\mathrm{d}}_a \geq \delta$, where δ is the maximum waiting time. Analogously, we can penalize waiting times gradually by increasing the travel cost function value proportionally to the waiting time.

5.3 Proposed Method

In this section, we describe the proposed method and prove that it solves the chaining problem optimally. The fleet-sizing article by Vazifeh et al. (2018) formulates the fleet-sizing problem as the *minimum path cover* (see Figure 5.1). Also, the authors conclude that the minimum path cover is equivalent to the maximum bipartite matching (see Figure 5.2) and, therefore, it can be solved in polynomial time by the Hoptcroft-Karp algorithm. Analogously, we can minimize the cost of the plan chains by replacing the maximum matching with the min-cost (perfect) matching, that is, by solving the assignment problem. This

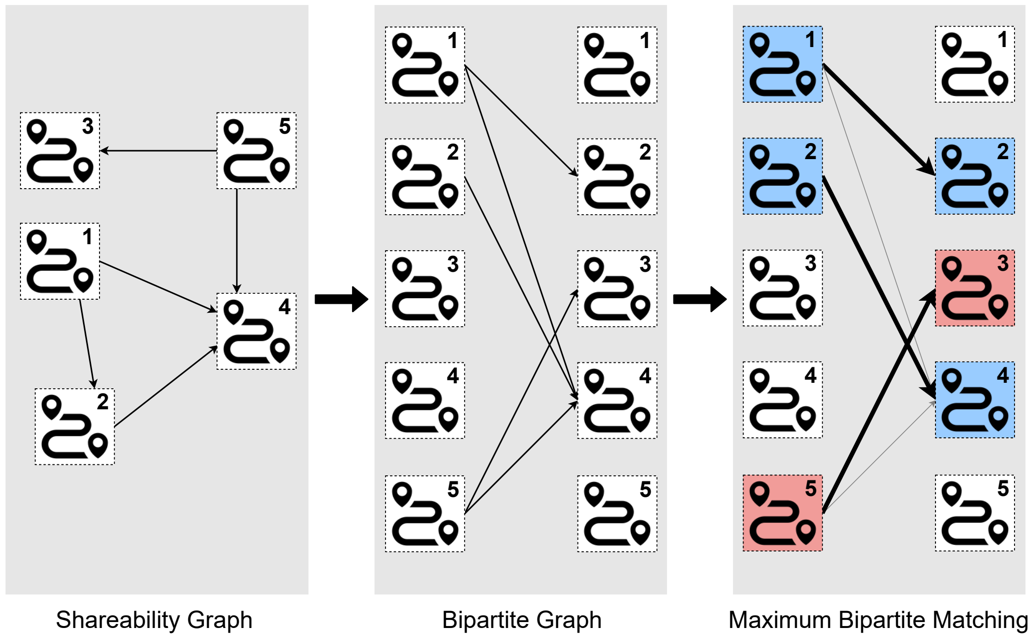


Figure 5.2: Example showing how the min path cover can be converted to the maximum bipartite matching. We use the same example as in Figure 5.1; plans are now numbered for clarity. Each color represents a chain of plans to be served by one vehicle. The connections (arrows) between plans in the chain are bold.

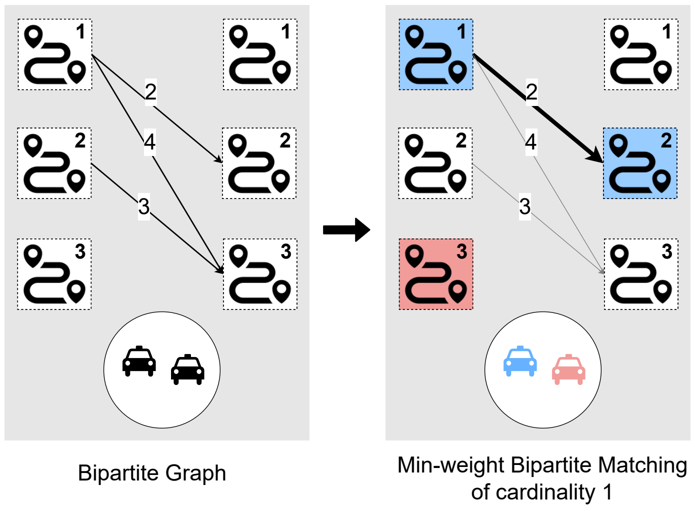


Figure 5.3: An example of plan chaining formulated as an assignment problem is a min-weight matching of a specific cardinality. The cardinality is given by the number of vehicles and plans: here, we have two cars available (below, in the circle) and three plans, resulting in cardinality 1 (at least one connection between plans). The numbers on the arcs determine the travel cost between the plans. Each color represents a chain of plans to be served by one vehicle. The connection (arrow) between plans in the chain is bold.

problem can also be solved in polynomial time (e.g., by the Hungarian Algorithm). An example of min-cost matching is shown in Figure 5.3. Note that, here, we also need to represent vehicles. For simplicity, the example assumes zero travel cost between the vehicle location and the start location of any plan.

However, even the assignment problem formulation is not suitable for optimal plan chaining, as it misses an important aspect: the DARP time windows. The delayed plans have a different shareability potential and, conclusively, delaying a plan results in a different bipartite graph. To solve the chaining Problem 5.1, we propose a two-step method. First, we generate only those delayed plan variants that are necessary to guarantee the optimal solution to the chaining problem. This variant generation process that generates only a fraction of all possible delayed plan variants is described in Section 5.3.1. Second, we formulate the chaining problem as a constrained min-cost flow problem (MCFP). This way, we can compute the optimal solution even for large instances despite the problem being NP-hard. In Section 5.3.2, we briefly introduce the min-cost flow problem (MCFP) formalization. Then, in Section 5.3.3, we present the constrained MCFP that represents the chaining problem and we show that an optimal solution of the proposed MCFP is an optimal solution of the corresponding chaining problem.

5.3.1 Generating Plan Variants

In the chaining formulation, we used a set of delayed plans Φ for each plan $p \in P$. In theory, time is a continuous quantity, so the number of delayed plan variants for each plan is infinite. However, in real-world computations, we usually consider a discrete time, which results in a finite number of plan variants. For example, with a resolution of one second and a plan that can be delayed by 20 seconds, there exist 20 (delayed) plan variants.

Nevertheless, as we prove further in this section, generating all possible plan variants is not necessary for guaranteeing the optimality of the plan chaining method. Instead, we generate only the variants that have the potential to extend the number of possible plan chains. The algorithm we propose for generating these variants is displayed in Algorithm 5.

Algorithm 5 Algorithm for generating plan variants.

```
Input: $P$: a set of plans, $V$: a set of vehicles
1: $\Phi \leftarrow \{\}$
2: $X \leftarrow \{\}$
3: variant_queue $\leftarrow$ empty queue

4: procedure TRY_CONNECT($a$: plan or vehicle, $b$: plan)
5:     if $a \in P \cup \Phi$ then
6:         min_delay $\leftarrow f_{\mathrm{tt}}(a,\ b) - (t^{\mathrm{o}}_b - t^{\mathrm{d}}_a)$
7:     else
8:         min_delay $\leftarrow f_{\mathrm{tt}}(a,\ b) - (t^{\mathrm{o}}_b - t^{\mathrm{s}}_a)$
9:     if min_delay $\leq 0$ then
10:        $X \leftarrow X \cup (a, b)$
11:    else if min_delay $\leq \delta^{\max}_b$ then
12:        $\phi \leftarrow$ delay_plan($b$, $min\_delay$)
13:        $X \leftarrow X \cup (a, \phi)$
14:        $\Phi \leftarrow \Phi \cup \phi$
15:        variant_queue.push($\phi$)

16: for $a \in P \cup V$ do
17:     for $b \in P \setminus a$ do
18:         TRY_CONNECT($a$, $b$)
19: while variant_queue not empty do
20:     $\phi \leftarrow$ variant_queue.pop()
21:     for $p \in P$ do
22:         if $\phi$ not variant of $p$ then
23:             TRY_CONNECT($\phi$, $p$)
```

Apart from generating variants, the algorithm also generates possible *connections* between vehicles and plans. We define the connection as an (ordered) pair (a, b), where $a \in P \cup V \cup \Phi$ and $b \in P \cup \Phi$. Plan or vehicle a can be connected to plan b only if: 1) a is a vehicle or the non-delayed version of a is different than the non-delayed version of b: $\Phi_a \neq \Phi_b$; and 2) the time difference between a and b is less than or equal to the travel time between them:

$$f_{\mathrm{tt}}(a,\ b) \leq \begin{cases} t_b^{\mathrm{o}} - t_a^{\mathrm{d}} & \text{if } a \in P \cup \Phi \\ t_b^{\mathrm{o}} - t_a^{\mathrm{s}} & \text{if } a \in V \end{cases} \tag{5.10}$$

The variant generation algorithm first tries to connect each plan and vehicle a with each plan $b \neq a$. If the connection is not possible, we compute a delay for plan b as

$$\delta_b = f_{\mathrm{tt}}(a,\ b) - \begin{cases} (t_b^{\mathrm{o}} - t_a^{\mathrm{d}}) & \text{if } a \in P \cup \Phi \\ (t_b^{\mathrm{o}} - t_a^{\mathrm{s}}) & \text{if } a \in V \end{cases}$$

If $\delta_b \leq \delta_b^{\max}$, we generate a delayed plan variant of b with delay δ_b and the corresponding connection.

Then, analogously, we try to connect each variant $\phi \in \Phi$ with each plan $p \in P$ and try to delay the plan p if necessary:

$$\delta_p = f_{\mathrm{tt}}(\phi,\ p) - (t_p^{\mathrm{o}} - t_\phi^{\mathrm{d}}),$$

Variant Generation and Plan Chaining Optimality

It is clear that Algorithm 5 does not generate all possible plan variants. However, we can prove that it generates all plan variants necessary to construct the optimal solution to the chaining problem.

Theorem 5.1. *There exists an optimal solution to the chaining problem that contains only non-delayed plans and plan variants generated by Algorithm 5.*

Before proving Theorem 5.1, we will analyze some properties of Algorithm 5. First, we assume that the connection cost does not depend on time:

Assumption 1. All plan chains that differ only in the delay of their plan variants have equal costs.

Next, note that when delaying the plan, the Algorithm 5 always uses the minimum possible delay (see function `try_connect`).

Lemma 5.2. *The function* `try_connect` *from the variant generation algorithm always creates the connection from plan o to plan p, if possible, using the minimum possible delay for plan p.*

Proof. We can prove this by inspecting the algorithm. On line 6, the minimum delay is computed as a difference between travel time between o and p ($f_{\mathrm{tt}}(o,\ p)$) and the time difference between these plans. We can see that this is indeed the minimum delay for a connection to be possible, as the delay is basically equal to the travel time minus the already existing time difference between plans. On the following lines, we can see that either:

- `min_delay` ≤ 0: no delay is needed, so we keep the plan p as is. As each original plan has the minimum possible delay (so we cannot assign a negative delay to a plan), this is indeed the minimum possible delay.
- $0 <$ `min_delay` $\leq \delta_p^{\max}$: in this case, we assign the minimum delay computed as described above.
- `min_delay` $> \delta_p^{\max}$: connection is impossible as the time constraints of plan p would be violated.

Thus, the claim is proved. □

Finally, note that for a sequence of plans to be a plan chain, all its sub-sequences have to be also plan chains, as stated in the following.

Lemma 5.3. *If we remove a plan from the end of a plan chain, the resulting sequence is also a plan chain.*

Proof. It trivially comes from Definition 5.1. □

Now, we can go back to Theorem 5.1:

Theorem 5.4. *An optimal solution to the chaining problem exists that contains only non-delayed plans and plan variants generated by Algorithm 5.*

Proof of Theorem 5.1. We will proceed by induction and show that for each possible plan chain, Algorithm 5 generates a set of connections that compose an equivalent plan chain that differs only in the plan delays (and in conclusion, has an equal cost, see Assumption 1).

Let us start with a chain of length one: a vehicle. As the algorithm does not discard any vehicles, it is clear that all chains of length one are covered. Let a chain of length two be given. Thus, we have a vehicle v and a plan p, and we try to connect them by a connection (v, p). There are two possible cases:

1. The connection (v, p) is not possible: we cannot generate the plan chain.
2. The connection (v, p) is possible: as we can see, all such connections are created by Algorithm 5 (lines line 18 and 23).

Analogously, the chains of length 3 (a vehicle and two plans) are generated by Algorithm 5. Let us now assume that Algorithm 5 generates all possible plan chains of length $k \leq n$, and suppose we want to connect chain $(p_1, \cdots, p_{n+1})$. On the account of Lemma 5.3, the problem boils down to connecting the chain $(v, p_1, \cdots, p_n)$ to (possibly delayed) plan p_{n+1}. There are three possible cases:

1. The connection (p_n, p_{n+1}) is not possible: we cannot generate the plan chain.
2. The connection (p_n, p_{n+1}) is possible: as we can see, all such connections are created by line 18 of Algorithm 5 in case that $\delta_{p_{n+1}} = 0$; by line 23 otherwise.
3. The connection (p_n, p_{n+1}) would be possible only with an earlier variant of p_n. However, according to Lemma 5.2 and by the induction hypothesis, we know that p_n is computed by the algorithm and is the earliest variant that can be connected to p_{n-1}. Therefore the chain $(v, p_1, \cdots, p_n, p_{n+1})$ is not possible.

Thus, the proof is complete. □

Complexity

When analyzing the complexity of Algorithm 5, we have the following assumption:

Assumption 5.1. The function `try_connect` has a constant time complexity.

With this assumption, we can only analyze the complexity of processing the `variant_queue`. We also build on the Assumption 2.2 about a constant ratio between requests and vehicles. Finally, let us assume that there is a minimum duration for any plan:

Assumption 5.2. For each $p \in P$, $t^{\mathrm{d}}_p - t^{\mathrm{o}}_p \leq \Delta$, where $\Delta \in R^+$

Below, we show that the complexity of the variant generation is finite and limited according to the following theorem:

Theorem 5.4. *If the Assumptions 2.2, 5.1, and5.2, the Algorithm 5 is finite and its complexity is:* $O\left(|P|^{\frac{\tau}{\Delta}+1}\right)$.

Proof. First, let us present the first few iterations of the algorithm. We initialize the `variant_queue` with all plans and vehicles. For each of those plans and vehicles, we try to connect them with all other plans and vehicles. In the worst case, after processing the initial non-delayed plans, we generate a new delayed plan variant for each plan combined with each other plan. The complexity, if the cycle is stopped after processing all non-delayed plans, is $O\left(|P|(|P|-1) + |V||P|\right)$. When going further, for each of these new delayed plan variants, we can, in the worst case, generate a new delayed plan variant for each plan, except the plan that generated the variant: $|P| - 1$ variants.

We can see that this processing of the `variant_queue` is potentially infinite. However, we will show that using Assumption 5.2 hold, the number of generated plan variants is finite. Let us divide the `variant_queue` processing into *generations*, i.e., the first generation tries to connect the plans among themselves, the second generation connects plans to the delayed plan variants generated in the first generation, and so on. Then, we can write the complexity as:

$$O\left(|P|(|P|-1)^{a}+|V||P|(|P|-1)^{a-1}\right), \tag{5.11}$$

where a is the number of generations. Looking at how `min_delay` is computed, we know that a variant ϕ' generated in generation i will start after the end of the variant ϕ generated in generation $i-1$: $t^{\mathrm{d}}_{\phi}+\delta < t^{\mathrm{o}}_{\phi'}$, even with the minimal plan delay $\delta \to 0$. Asumming 5.2 is true, we can now bound the maximum number of generations by $\frac{\tau}{\Delta}$, where τ is the time horizon of the DARP instance as defined in Chapter 2. The complexity of the variant generation algorithm is then

$$O\left(|P|(|P|-1)^{\frac{\tau}{\Delta}}+|V||P|(|P|-1)^{\frac{\tau}{\Delta}-1}\right). \tag{5.12}$$

Using Assumption 2.2, we can write the complexity as:

$$\begin{aligned} &O\left(|P|(|P|-1)^{\frac{\tau}{\Delta}}+|P|^{2}(|P|-1)^{\frac{\tau}{\Delta}-1}\right)\\ =\;&O\left((|P|)^{\frac{\tau}{\Delta}+1}+|P|^{\frac{\tau}{\Delta}+1}\right)=O\left((|P|)^{\frac{\tau}{\Delta}+1}\right)\end{aligned} \tag{5.13}$$

□

From the analysis above, the following takeaways can be made. First, if we consider only realistic plans that have a minimum duration, the complexity (and the number of generated plan variants) is finite. Moreover, the complexity is polynomial relative to the input: the exponent depends on the parameters of the problem rather than the size of the input. This is an important consideration for estimating which problems can be solved by this chaining method. Large problems may be easy to solve, while some small problems with a large time horizon can be unsolvable in practice. This is a similar situation to the VGA method, which also fails to solve some problems with a large time horizon. Therefore, it may seem that the effort to deliver a method able to solve these hard instances was fruitless. Nevertheless, as the results (Section 5.4) show, the optimal chaining can be applied to much longer time horizons than the VGA method.

Finally, we should remark that the practical complexity is much lower than the above presented worst-case complexity, as a) the maximum delay of plans is usually much smaller than the time horizon, and b) a considerable time is usually needed to travel between plans, which effectively limits the number of connections further.

5.3.2 Min Cost Flow Problem Formalization

The minimum-cost flow problem (MCFP, see Ahuja et al. (1993)) is an optimization problem of finding a minimum set of flows through a flow network that is in total equal to a specified amount of flow.

Definition 5.2. A *flow network* is a directed graph $G = (K, E)$. Each edge e from E is defined as a 4-tuple (f_e, l_e, u_e, c_e), where:

- f_e is the flow of the edge e, a variable,
- l_e is the lower bound of the edge e, a constraint,
- u_e is the upper bound of the edge e, a constraint,
- c_e is the cost of the edge e, a constant.

Each vertex κ from a set K has an associated supply value s_κ.

If $s_\kappa > 0$, the node is called a *source*, if $s_\kappa = 0$ it is a *transshipment node*, and if $s_\kappa < 0$ it is called a *sink*. We mark the set of edges originating in the vertex κ as O_κ and a set of edges with a destination in κ as D_κ.

The min-cost network flow problem is then formulated as:

Problem 5.2. Minimize:

$$\sum_{e \in E} c_e f_e, \tag{5.14}$$

subject to

$$\sum_{e \in O_\kappa} f_e - \sum_{e \in D_\kappa} f_e = s_\kappa \quad \forall \kappa \in K, \tag{5.15}$$

where $f_e \in [l_e, u_e] \quad \forall e \in E$.

Here, Equation 5.15 is the flow conservation constraint.

5.3.3 Chaining as the Min-cost Flow Problem

To find an optimal solution to the chaining problem given the vehicles, plans, and delayed plan variants and connections generated by Algorithm 5, we propose a constrained min-cost flow problem formulation exemplified in Figure 5.4.

There are seven types of nodes:

- A single source and a single sink node,
- left and right plan nodes for each original plan,
- left and right variant nodes generated for plans with delayed variants.
- vehicle nodes

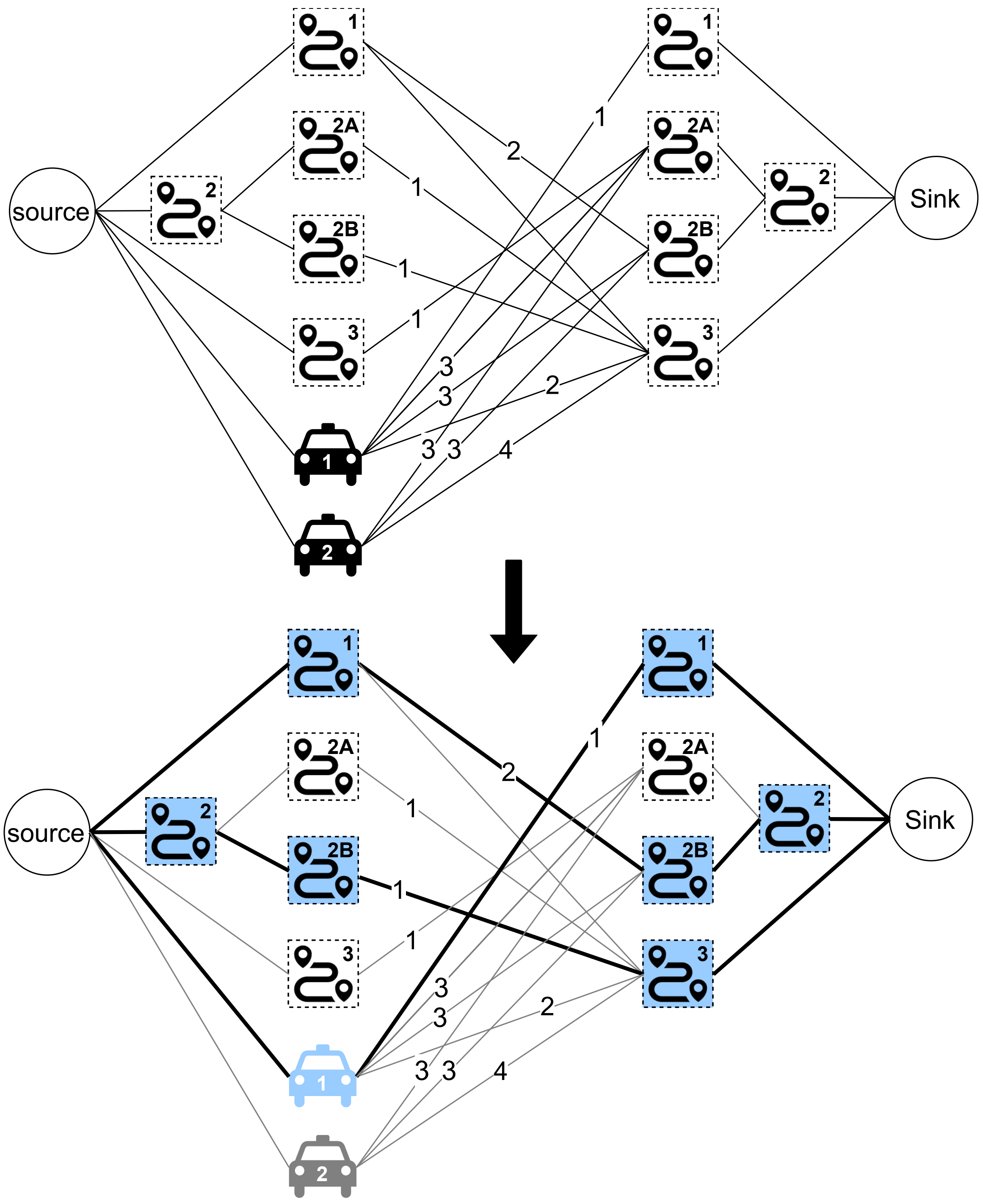


Figure 5.4: Example of the final chaining formulation formulated as an MCFP. The vertices are, from the left: the source, then left plan vertices for each plan, left variant vertices for each variant (e.g., 2B translates to plan 2, variant B), and vehicle vertices, then right plan and variant vertices, and finally, the sink. If a plan has no delayed variants, there is no variant vertex. The travel cost is expressed by the number over the arc. Arcs without numbers have zero cost. The inflow of the source is equal to the number of plans, and the outflow of the sink is reversed to that. In the bottom image, there is the solution marked by bold arcs (used arcs with active flow). For readability, vertices between solution arcs are painted blue.

Each node has a zero supply, with the exception of the source node, which has a supply equal to the number of plans (three in the example), and the sink node, which has the inverse supply. The edges are generated:

- from source to each left plan node and each vehicle node,
- from each left plan node to each of its corresponding variant nodes,
- from each right variant node to its corresponding right plan node
- from each right plan node to sink,
- and from vehicles and left plans/variants to right plans/variants according to the output of the Algorithm 5.

Each edge has a direction from left to right. Also, for each edge e, $l_e = 0$, and $u_e = 1$. The cost of each edge is zero, with the exception of the edges between left plan/variant nodes and right plan/variant nodes that have a cost equal to the travel time between the destination of the previous plan and the origin of the next plan.

To solve the chaining problem optimally, we solve this min-cost flow problem and then 1) connect (chain) plans for each edge between the left plan/variant node and the right plan/variant node with an active flow (flow = 1), and 2) assign vehicles to chained plans according to the active flows outgoing from the vehicle nodes. For example, in the solution illustrated in Figure 5.4 by bold lines, the edges (vehicle 2, plan 1), (plan 1, plan 2B), and (plan 2B, plan 3) are part of the solution. Therefore, the resulting solution is a single plan chain (vehicle 2, plan 1, plan 2B, plan 3).

To guarantee that this process results in a feasible system plan, we need to constrain the min-cost flow problem such that a) only one variant has an active connection on the left part of the flow problem, b) only one variant has an active connection on the right part, and c) the connected variant on the left is the same as the one connected on the right. Conveniently, conditions a) and b) are guaranteed by the flow conservation constraint. Next, we have to introduce a constraint to guarantee the condition c). We need two symmetric constraints:

$$f(\kappa_l^p, \kappa_l^\phi) + \sum_{\substack{\phi' \in \Phi_p \\ \phi' \not\equiv \phi}} f(\kappa_r^{\phi'}, \kappa_r^p) \leq 1 \quad \forall p \in P, \forall \phi \in \Phi_p \tag{5.16}$$

$$f(\kappa_r^\phi, \kappa_r^p) + \sum_{\substack{\phi' \in \Phi_p \\ \phi' \not\equiv \phi}} f(\kappa_l^p, \kappa_l^{\phi'}) \leq 1 \quad \forall p \in P, \forall \phi \in \Phi_p \tag{5.17}$$

Here κ_l^p is the left plan vertex connected to the left variant vertex κ_l^ϕ, and κ_r^p is the right plan vertex connected to the right variant vertex κ_r^ϕ, As these constraints are generated for each variant node, there are no such constraints for plans without delayed plan variants.

From the description above, it results that the proposed MCFP corresponds to problem 5.1. Therefore, by optimally solving the MCFP, we obtain an optimal solution to the chaining problem.

Note that with the additional constraints, we lose the ability to use the LP solver because the flows could now be non-integer. Therefore, we have to solve this formulation using an ILP solver, which is an NP-hard problem. In line with Section 2.3.2, will not analyze the complexity of the ILP solver algorithm, but we will try to estimate the bound of the ILP problem size. The number of binary flow variables can be computed as a sum of:

- edges from the source to vehicles: $|V|$,
- edges from vehicles to plans and variants. This can be bounded by: $|V|(|P| + |\Phi|)$.
- edges from the source to plans and from plans to sink: $2|P|$,
- edges from the plans to the variants and from the variants to the plans. This can be bounded by: $2|\Phi|$,
- connection (chaining edges): $(|P| + |\Phi|)(|P| + |\Phi| - 1)$.

Next, the number of flow conservation constraints is equal to the number of vertices, which is $|V| + 2(|P| + |\Phi|) + 2$. Finally, there are two variant conservation constraints (5.16, and 5.17) for each variant: $2|\Phi|$.

5.4 New DARP Heuristic Method Based on Optimal Plan Chaining

In this section, we demonstrate how the proposed optimal chaining method can be used to solve DARP. Specifically, we focus on DARP instances with long time horizons, where the optimal solution is computationally intractable. We start by introducing the problem and the scheme of the proposed method based on optimal chaining (Section 5.4.1). Next, we present the formal changes to the VGA method compared to the formulation presented in Section 2.3.2 (Section 5.4.2). Then, in Section 5.4.3, we described briefly the instance and methods used in this demonstration, and finally, Section 5.4.4 presents the results.

5.4.1 Principle of the Method

As we show in the previous Chapter, one of the optimal methods for solving the ridesharing DARP, the Vehicle-group assignment (VGA) method (Alonso-Mora et al., 2017), suffers from high computational complexity when solving instances with long time horizon (more than tens of seconds). Therefore, when solving instances spanning a longer time (for example in offline ridesharing), a different method is required. In this demonstration, we propose a new heuristic method that uses the VGA method as its component. Unlike classical metaheuristics (see review in Ho et al. (2018)), which are iterative methods, the heuristic proposed here is constructive: it creates a single solution and does not improve it afterward. However, existing metaheuristics can be initialized by the solution created by the proposed heuristic method as we demonstrate in the results section.

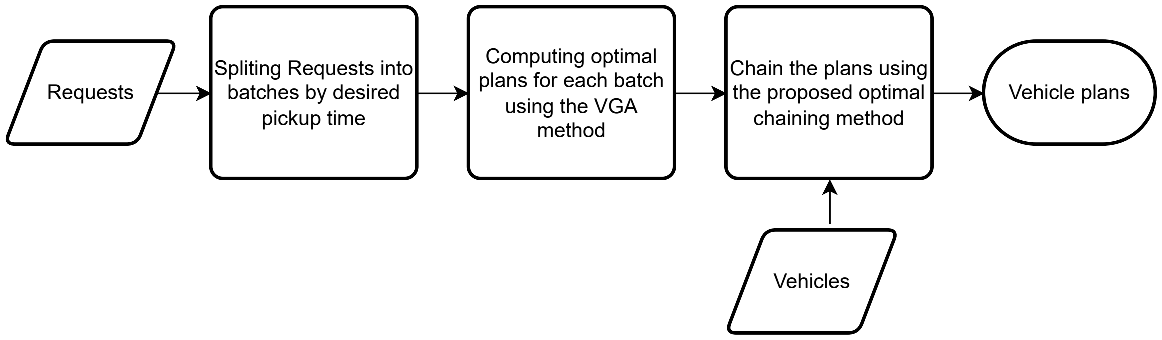

Requests
Spliting Requests into batches by desired pickup time
Computing optimal plans for each batch using the VGA method
Chain the plans using the proposed optimal chaining method
Vehicle plans
Vehicles

Figure 5.5: The scheme of the proposed method.

The principle of the proposed heuristic method is to split the demand by time into short time intervals (*batches*) that can be solved by the VGA method optimally. This is similar to the classic VGA method operating in an online mode. Here, however, we know the content of the batches in advance, so we can solve them simultaneously. Then, these batches are joined by the optimal chaining method proposed in this chapter, resulting in plans that cover the whole instance period. The simplified scheme of the method is in Figure 5.5. Note that in this scheme, the VGA method can be replaced by any other optimal DARP solution method.

5.4.2 Formal Changes to the VGA Method

The main difference from the original formulation is that, here, we cannot use the initial position of vehicles, as we do not know it as a result of splitting the demand into batches (except for the first batch) Therefore, instead of real vehicles, we use *virtual vehicles*. The real vehicle assignment is done later in the chaining procedure. For group generation (see Section 2.3.2) this affects the function $f(G, v)$ which determines if a group G can be served by vehicle v. Instead of the travel time from the vehicle position to the pickup location, we use a single constant time t_s, an estimate of travel time from the vehicle (unknown) position to the origin of any request. This estimate is the parameter of the algorithm.

Because the constant t_s is equal for all vehicles, we can simplify the VGA method further, affecting the vehicle-group assignment (see Section 2.3.2) as well. We consider just a single virtual vehicle for the whole problem. This way, we can run the group generation only for one virtual vehicle. Next, in the vehicle-group assignment, instead of a maximum of one plan per vehicle, we allow the virtual vehicle to serve the number of plans equal to the number of real vehicles in the system. The modified vehicle constraint for the vehicle-group assignment looks as follows:

$$\sum_{R \in \gamma} x_R = |V| \tag{5.18}$$

Table 5.1: Used DARP instances and their parameters.

Instances	Duration	Requests	Vehicles	Trip Length [mean±std min]
DC	5 min	54	50	15.2±9.8min
DC	15 min	163	121	15.8±8.7min
DC	30 min	328	180	15.9±8.4min
Chicago	5 min	91	65	14.2±16.6min
Chicago	15 min	274	198	13.4±15.1min
Chicago	30 min	596	299	14.1±15.9min
Manhattan	5 min	1658	781	7.8±4.0min
Manhattan	15 min	5113	1672	7.7±3.8min
Manhattan	30 min	10362	1993	7.6±3.8min
NYC	5 min	3488	2384	10.1±6.8min
NYC	15 min	10567	5085	9.9±6.6min
NYC	30 min	20841	5905	9.9±6.5min

5.4.3 Instances, Evaluated Methods, and Experiment Configuration

For the demonstration, we used large-scale ridesharing DARP instances presented in Section 3.2. These instances are located in four areas (New York City, Manhattan, Chicago, and Washington, DC) that vary not only geographically but are also very different in the magnitude of both road network and travel demand (see Section 3.2.2). For each area, we selected only instances with a maximum delay of 5 minutes and instance duration of 5, 15, and 30 minutes. We consider the request to be served by conventional personal vehicles and, therefore, we set the vehicle capacity to four persons. The parameters of all used instances are displayed in Table 5.1

We compared the proposed heuristic method with three other methods for solving DARP, each representing a single category of DARP solution methods:

- the vehicle Vehicle-group Assignment (VGA) method (Alonso-Mora et al., 2017; Čáp & Alonso-Mora, 2018): an optimal solution method,
- the Insertion Heuristic (IH) (Jaw et al., 1986), implemented as in Fiedler, Čertický, Alonso-Mora, et al. (2022), representing simple construction heuristics,
- and the Adaptive Large Neighborhood Search (ALNS) (Ropke & Pisinger, 2006), configured as in Masmoudi et al. (2020)[1].

[1]We omitted the “hybrid” part as implementing the genetic operators for free-floating vehicles would be too complicated, considering that we use this metaheuristic only as one of three baseline methods.

As the selected set of instances is challenging for the solution methods due to the large scale, we cannot expect that all the solution methods will be able to compute all instances. However, we include even the optimal method with exponential complexity to cover all major types of DARP solution methods. Considering the proposed method, we tested multiple of its configurations. First, we tried batch lengths of 30, 6, 120, 240, and 480 s. Second, we also tried a time-limited version where the underlying VGA method does not solve the individual batches optimally but is time-limited instead.

We implemented all methods in C++. For solving mixed integer programs, we used the Gurobi (Gurobi Optimization, LLC, 2023). We ran the experiments on the AMD EPYC 7543 CPU. The computation was limited to 24 hours. Because some of the methods also require an extensive amount of memory, we limited the RAM usage to 80 GB, not including the memory used to store the distance matrix for travel time computation (which varies dramatically between areas). As the proposed method can be easily parallelized, we used multiple threads to solve the method. Note, however, that our parallelization is limited to the trivially parallelizable algorithms (computing batches in parallel) or the library calls with built-in parallelization (Gurobi solver). We believe that the evaluated methods can be further parallelized to achieve a smaller computational time. Finally, we did not spend extensive time with software optimizations of the evaluated methods. Therefore, the computational time results should be taken with a grain of salt.

5.4.4 Results

We present the main results in four tables: one for each area. The areas are discussed from the most complex (largest by demand). In each area table, only the methods that were able to solve at least one instance for the area are included.

Table 5.2 shows the results for New York City. Only the variants of the chaining method, together with IH, were able to solve any instance within the time limit. Moreover, some of the variants of the proposed method were not able to solve the instance either, and the 30 minute instance was solved only by the IH. However, for the remaining two instances, the proposed method provides the best results.

Similarly, in the Manhattan instance, (Table 5.3), the proposed method beats the IH in 2 of 3 instances. Here, the proposed method was able to compute even the instance with the longest time horizon; however, the successful variants provide worse results than IH. The ALNS variants were able to compute at least the 5 minute instance here, but they were outperformed by the proposed method. Note, however, that the ANLS-VGA variant is outperformed only by the proposed method with the batch length of 480 s which actually does not chain anything, as the batch is longer than the instance time horizon. The reason why this method is able to compute the instance, while the VGA is failing is probably because the proposed method computes the VGA part without considering the positions of the vehicles, even for the first batch. This leads to suboptimality, but also to the reduction of CPU time and memory requirements.

In the Chicago area, the shortest instance was solved to optimality, showing the limits for practical application of the heuristic methods. All other methods were able to solve all instances, with the exception of the 30 and 60 s variants of the proposed method. The best

Table 5.2: NYC results. None of the NYC instances were solved by the optimal method in time. The same is true for the metaheuristic variants

			Total Cost [min]			**Used Vehicles**			**Comp. time [s]**		
Method	Batch [s]	Lim.	5 min	15 min	30 min	5 min	15 min	30 min	5 min	15 min	30 min
IH	-	no	30397	81486	**157023**	1613	**3716**	**5031**	**5.76**	**39.55**	**91.87**
Proposed met.	30	no	34607	100729	-	2061	4812	-	61.68	1004.12	-
	60	no	31789	93646	-	1854	4572	-	43.26	709.80	-
	120	no	29364	85486	-	1680	4269	-	249.57	897.20	-
		yes	29372	85487	-	1680	4266	-	135.45	719.93	-
	240	yes	27509	-	-	1567	-	-	224.84	-	-
	480	yes	**25736**	**76832**	-	**1450**	4308	-	223.54	1020.74	-

Table 5.3: Manhattan results. Here, almost all variants of all methods provide results for at least one instance. The ALNS-IH method is ALNS initialized with the solution of the insertion heuristic. The ALNS-prop is the ALNS initialized with the proposed method of batch 120 s.

			Total Cost [min]			**Used Vehicles**			**Comp. time [s]**		
Method	Batch [s]	Lim.	5 min	15 min	30 min	5 min	15 min	30 min	5 min	15 min	30 min
ALNS	-	no	8533	-	-	516	-	-	28891.51	-	-
ALNS-IH	-	no	8371	-	-	512	-	-	25707.88	-	-
ALNS-prop	-	no	7744	-	-	**505**	-	-	27655.04	-	-
IH	-	no	9626	25386	**48420**	594	**1306**	**1721**	**0.61**	**6.09**	**18.40**
Proposed met.	30	no	10426	-	-	738	-	-	3.96	-	-
	60	no	9140	27686	55533	635	1648	1993	3.45	47.90	1706.07
	120	no	8312	24360	48662	568	1504	1963	144.69	273.49	845.55
		yes	8305	24360	48662	568	1504	1963	107.69	274.43	835.12
	240	yes	7764	**21646**	-	547	1383	-	205.15	625.21	-
	480	yes	**7445**	-	-	562	-	-	205.90	-	-

Table 5.4: Chicago results. Here, almost all methods compute all the instances. The ALNS-IH method is ALNS initialized with the solution of the insertion heuristic. The ALNS-prop is the ALNS initialized with the proposed method of batch 120 s.

			Total Cost [min]			**Used Vehicles**			**Comp. time [s]**		
Method	Batch [s]	Lim.	5 min	15 min	30 min	5 min	15 min	30 min	5 min	15 min	30 min
ALNS-IH	-	no	1165	2998	**6612**	42	**97**	**166**	57.58	699.08	10049.92
ALNS-prop	-	no	1153	**2982**	6690	**41**	102	170	56.85	723.35	10467.21
IH	-	no	1200	3117	7090	46	114	209	**0.01**	**0.04**	**0.15**
VGA	-	no	**1129**	-	-	42	-	-	15.14	-	-
Proposed method	30	no	1451	3940	-	57	165	-	0.12	0.62	-
	60	no	1411	3825	-	55	153	-	0.11	0.40	-
	120	no	1311	3613	8943	51	142	258	0.14	0.32	1.84
		yes	1311	3613	8943	51	142	258	0.11	0.35	1.84
	240	no	1234	3334	8246	48	129	240	0.12	0.42	1.38
		yes	1234	3334	8246	48	129	240	0.13	0.46	1.35
	480	no	1132	3107	7391	**41**	112	212	0.36	10.05	32.18
		yes	1132	3107	7391	**41**	112	212	0.37	10.15	32.21

solution for the 15 minutes and 30 minute instances, is provided by ALNS. For the 15 minute instance, however, it is the ALNS variant initialized by the proposed method.

In the DC instance, we were able to solve all instances optimally. Therefore, comparing the heuristic methods is irrelevant. However, we can observe that for the 5 minute instance, the cost of the heuristic solutions of some methods is very close to the optimal solution. This signalizes that when the solution space is smaller, the heuristic methods perform well compared to the more complex instances.

If we look at the computational time, it is clear absolute winner is the insertion heuristic. We can see that with an increased instance time horizon, the computational time grows more than linearly. Compared to it, the optimal method is slower, and the computational time grows even faster, exponentially with the instance time horizon. When we compare VGA and the proposed method, the proposed method is mostly faster, and the computational time does not grow that fast. However, we can see differences between method variants: longer batches result in a linear computation time growth, while shorter batches slow down faster. This is a result of the chaining, which is not parallelized, and is much more difficult when the batches are short. Finally, the metaheuristic is the slowest one, by far. It could be probably sped up by reducing the number of iterations but with the cost of reducing the quality of the solutions.

The last measured quality in the main result table is the number of used vehicles. This represents the capital cost of the solution, and also somehow indicates the vehicle occupancy, which is plotted separately in Figure 5.6 in the form of a histogram. To make the histogram

Table 5.5: DC results. The ALNS-IH method is ALNS initialized with the solution of the insertion heuristic. The ALNS-prop is the ALNS initialized with the proposed method of batch 120 s.

			Total Cost [min]			**Used Vehicles**			**Comp. time [s]**		
Method	Batch [s]	Lim.	5 min	15 min	30 min	5 min	15 min	30 min	5 min	15 min	30 min
ALNS	-	no	1062	2843	5252	39	85	127	19.21	149.25	979.12
ALNS-IH	-	no	1023	2789	5157	**38**	88	132	18.31	137.85	843.49
ALNS-prop	-	no	1029	2789	5159	**38**	87	129	18.25	150.52	1002.02
IH	-	no	1049	2934	5593	39	93	146	**0.00**	**0.01**	**0.04**
VGA	-	no	**1009**	**2677**	**4821**	**38**	**82**	**122**	0.09	2.31	457.56
Proposed method	30	no	1136	3545	6859	46	121	180	0.11	0.27	1.51
	60	no	1122	3496	6744	45	120	180	0.10	0.36	1.29
	120	no	1095	3373	6400	44	115	170	0.08	0.22	1.05
		yes	1095	3373	6400	44	115	170	0.06	0.20	1.05
	240	no	1023	3263	6095	39	113	169	0.06	0.22	0.61
		yes	1023	3263	6095	39	113	169	0.09	0.23	0.72
	480	no	1015	2983	5540	**38**	104	154	0.09	0.18	0.44
		yes	1015	2983	5540	**38**	104	154	0.08	0.18	0.43

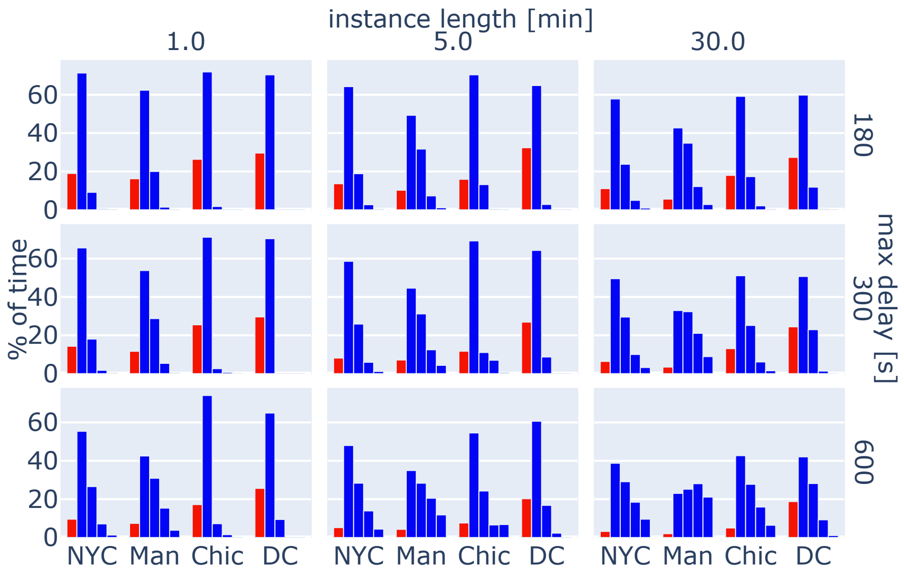


Figure 5.6: Occupancy histograms for all city-exp. length combinations. The red bar represents zero occupancy, and then the occupancy grows to the right up to the maximum occupancy of four passengers.

clear, we included only one variant of the proposed heuristic and one ALNS variant. In general, we can see that the demand in the Chicago and DC areas is not dense enough to make significant savings by ridesharing. The vehicle hours with occupancy of two requests are compensated by empty vehicle hours and only a small fraction of vehicle hours are driven with more than two requests on board. However, in the NYC area, the effect of ridesharing is already significant, and in Manhattan, we can see a high proportion of high occupancy, even fully occupied vehicles. When comparing the occupancy between individual methods, we can see a correlation with the solution cost: the methods with lower solution costs tend to have higher occupancy.

We also examined delays for each individual travel request. The histogram of the delays is in Figure 5.7. In general, the delays are higher for the methods that offer lower operational costs, as the increased share of shared trips causes more delays. However, this relationship is not strict. In the many histograms, we can observe that the IH and ALNS cause the highest delays, but the most efficient is the proposed method, or VGA (compare, e.g., to Table 5.3).

Overall, it is evident that we cannot choose a method that is best for all the instances. Rather than that, one message of these results, also previously indicated in Fiedler and Mrkos (2023a), is that it is essential to test the DARP methods on multiple instances with different characteristics, before making any assumptions on the best method. For small instances with low computational complexity, we can use the optimal method, even if the time horizon is long. For the hardest instances, only the IH is applicable. Finally, for the remaining instances with medium complexity, the proposed method is usually the best.

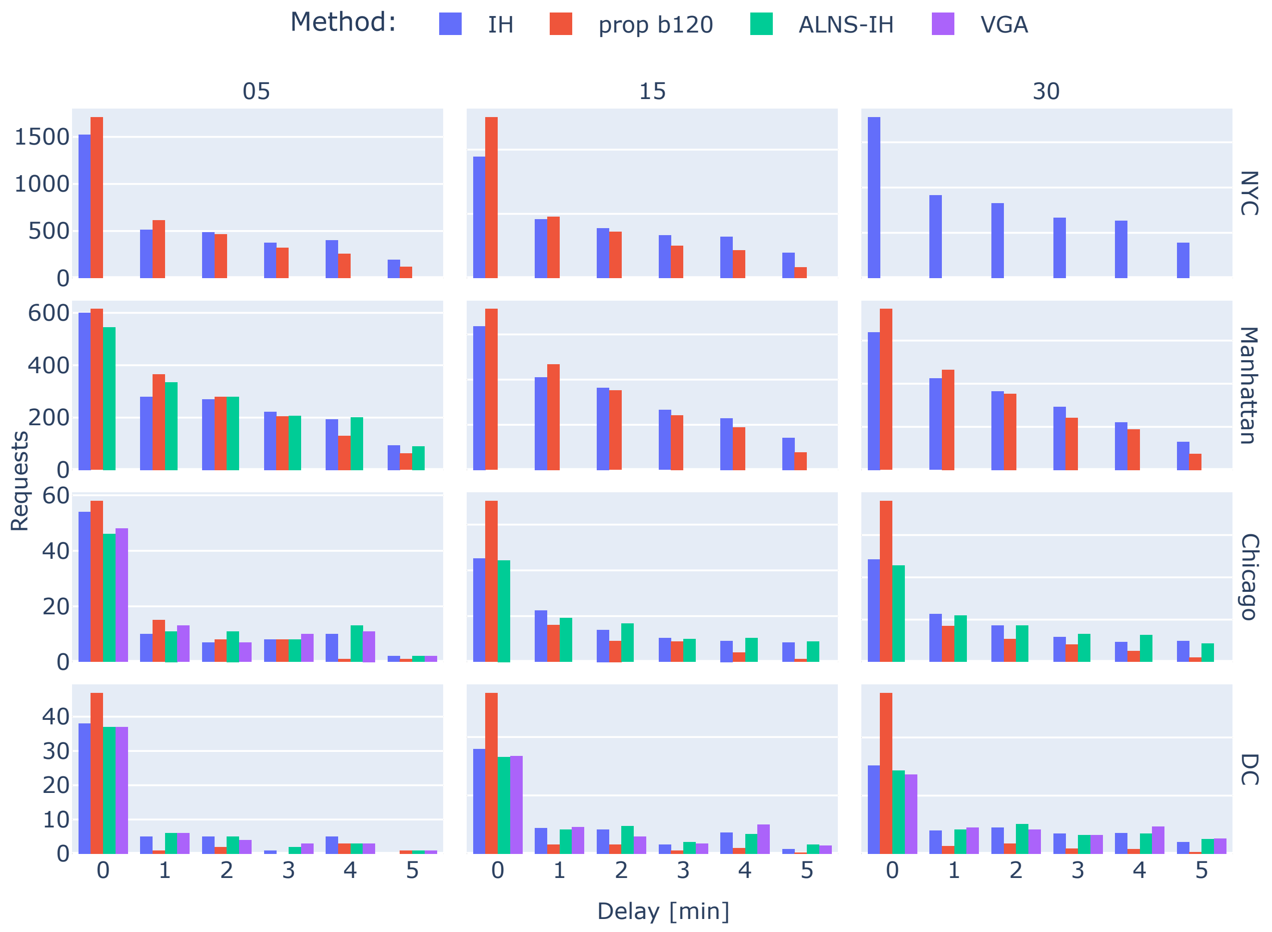


Figure 5.7: Delay histograms for all city-experiment length combinations. The `prop b120` method is the proposed method with a batch length of 120 s.

Note also that the ALNS-prop method, which sometimes outperforms all configurations of the proposed method, is initialized with the solution of the proposed method. Therefore, we think that this demonstration of the possible application of the optimal chaining proves its practical usability, and should be in the toolbox of future researchers focused on DARP.

5.5 Summary

In this chapter, we address the last research goal (RG5): to deliver a new heuristic method for solving the DARP. Specifically, we build on the findings of our simulation studies (Chapter 4) and focus on DARP instances that cannot be solved optimally in a reasonable time and at the same time, have a large gap between the optimal and heuristic solutions (which we can estimate from the sensitivity analysis trends). Based on these findings and related work, we propose a new heuristic method for solving the DARP. The principle of the method is to split the demand into short time intervals, solve them optimally, and then chain the plans. The plan chaining is known to be a subproblem in MoD literature, namely in the context of fleet sizing. However, after researching the work related to plan chaining, we found that the chaining problem in literature is usually solved with fixed plan times, regardless of the flexibility coming from the time windows. Typically, each travel request has some time flexibility, called a time window. These time windows propagate to the vehicle plans. Therefore, we provide a new formulation of the plan chaining problem with time windows. Next, as the success of the method depends on the quality of the chaining, we focused our efforts on delivering an optimal chaining method. We propose a solution method for solving the chaining problem, significantly reducing the search space compared to naive solutions. Moreover, we prove that the method is optimal. Finally, we evaluate the proposed DARP heuristic on large-scale DARP instances presented in Chapter 3. The evaluation results confirm the applicability of the proposed method, as it can solve instances that cannot be solved optimally within the time limit while being better than other evaluated heuristics in most of the evaluated instances. Moreover, it demonstrates that the optimal chaining method we have developed is practically applicable, and therefore, it opens a new research direction in using it in other MoD problems like fleet sizing.

CHAPTER 6
Conclusion

In the research summarized in this thesis, we studied mobility-on-demand (MoD) systems, with the main focus on the operation of large-scale MoD systems operated by transportation network companies (TNCs). The general conclusion of this thesis is that we were successful in fulfilling the goals we have set (see Section 1.1). The detailed results and conclusions of each research question are summarized at the end of the corresponding chapter (see Table 1.1). In the following section, we briefly summarize the main contributions of this thesis, organized by the goals they address. Then, in Section 6.2, we discuss the possible directions for future research.

6.1 Summary of the Contributions and Results

For analyzing MoD systems or developing new methods and algorithms for them, we first need to obtain **input data**. The data instances for MoD or, in general, mobility contain transportation demand, vehicles, and the travel time model. As we show in Chapter 3, there were no standardized instances for large-scale studies available in the literature. Either the instances are too small, uniformly generated and have different characteristics than the current MoD systems operated by transportation network companies (TNCs), or they are not standardized, and therefore, not comparable or reproducible. To address this issue, we have developed a methodology for creating large-scale instances based on historical travel data. Using this methodology, we have created and published a set of instances for three areas: New York City, Chicago, and Washington D.C. We tested the instances by solving the dial-a-ride problem (DARP) and showed that they are suitable for large-scale studies. The results of this evaluation also show the importance of using instances from various areas, as different algorithms perform better in different areas.

There are many research questions about MoD systems that were still unanswered when we started our MoD research. We selected three of them and answered them using multi-agent **simulations**. For all three research questions, we used the same simulation methodology and input data. We performed the simulations in two different areas: Prague and Manhattan. This simulation research is described in Chapter 4.

MoD systems remove the burden of personal car ownership and driving and have a clear advantage of reducing the required fleet size and parking requirements. However, they also produce empty trips, which are trips between one passenger's drop-off location and the next passenger's pick-up location. Before the research summarized in this thesis, it was not clear what is the trade-off between the reduction of the fleet size and the increase in the number of empty trips. Our simulation results revealed this trade-off. While the fleet size can be reduced 4 to 6 times, the total distance driven by the fleet increases by 20 % to 30 %.

To mitigate the increase in traveled distance, we can employ ridesharing, i.e., transporting multiple passengers in one vehicle simultaneously. However, it is not clear how much ridesharing can reduce the total distance driven by the fleet in a real-world MoD system. With the help of our simulation, we were able to quantify the benefits of ridesharing. We found that the total distance driven by the fleet can be reduced by 22 % compared to using private cars and by 46 % compared to the MoD system without ridesharing. Moreover, there is an additional reduction in the required fleet size compared to the MoD system without ridesharing: in general, we only need half of the vehicles.

The last question we answered using simulation is how much we can benefit from dispatching the vehicles optimally, i.e., by finding an optimal solution to the Dial-a-Ride Problem (DARP). This is an important question, as computing the optimal solution is computationally expensive and even infeasible for some instances. Therefore, trying to find the optimal solution only makes sense if the reduction in the cost (total distance driven) is significant. With the simulation, we were able to quantify the benefits of optimal ridesharing and we found that the gap between the optimal and heuristic solutions is significant. In the case studies, the optimal ridesharing reduces the total distance traveled by about 20 % compared to the Insertion Heuristic, while simultaneously reducing the average passenger delay by 5 %. Moreover, we also developed a resource-constrained variant of the optimal method and explored various trade-offs between the total distance traveled and the computational requirements.

The final goal of the thesis was to develop a new heuristic **optimization** algorithm for vehicle dispatching which translates the optimal solution of the DARP. In our simulation studies, we identified the weak point of the optimal solution method: the method was failing on instances with a long time horizon, i.e., with travel requests spread over a long time. To produce good solutions for these instances, we developed a new heuristic algorithm that splits instances with a long time horizon into smaller sub-instances, solves them optimally, and then chains the optimal solutions together. As there was no suitable chaining method in the literature that considers time windows, we developed such a chaining method and proved that it is optimal. We evaluated the new DARP heuristic on the instances developed for this thesis (RG1) and it proved to be very effective. It can solve instances that cannot be solved optimally in the 24 h time limit while being better than other standard heuristics in most of the evaluated instances. These results not only show the effectiveness of the DARP heuristic developed to address the RG5 but also the quality and applicability of the underlying optimal plan chaining method. As plan chaining is a subproblem of other MoD problems, this opens the research opportunities for the application of the optimal plan chaining method in other MoD problems.

6.2 Future Work

Although we have developed new methodologies and algorithms for MoD systems, and we have answered several research questions, there are still many research opportunities in this area.

The first opportunity is to continue the work on the DARP instances. To maintain the standardization, it is desirable to limit the number of instances and areas but still, we believe that there is room for more variety. The instances we have developed cover large cities in the US, but other countries and continents, potentially with different travel behaviors, are not represented. Also, different types of areas, such as suburban or rural areas can be of interest. Another aspect of the instances that can be improved is the travel time model. For that, we use the UberMovement dataset[1]. However, this dataset covers only a few areas and as of October 1, 2023, it is no longer available. Therefore, it is desirable to find or develop new travel time models.

The next opportunity is to explore various extensions or specializations of MoD systems that are yet extensively studied but represent real-world scenarios and trends. The first extension is the integration of parcel or food delivery into the MoD systems. Although there is some research in this area (Febbraro et al., 2018; Godart et al., 2018; Manchella et al., 2021), the modification and evaluation of state-of-the-art dispatching algorithms for this scenario are still missing. The second extension is the integration of public transport into the MoD systems. In spite of a great amount of research in this area (e.g., Feng et al., 2022; Lau and Susilawati, 2021; T.-Y. Ma et al., 2019), there are still some open questions, for example, which public transport lines should be replaced by MoD systems to provide a more comfortable and more efficient service? The third extension is enabling transfers between vehicles in MoD systems. This can reduce the required fleet size and enable a high level of service even in areas with low demand or penetration of MoD systems. Again, there is an existing body of research on this topic (Lotfi & Abdelghany, 2022; Lyu & Yu, 2023; Pierotti & van Essen, 2021; Voigt & Kuhn, 2022), but the studies are limited to small-scale instances. Another extension can be replacing the relaxation of the door-to-door operation, i.e., allowing the passengers to walk to the pick-up or from the drop-off location. This healthy and environmentally friendly option can be beneficial especially when ridesharing is employed, and it can, additionally to traveled distance, also reduce the requirement for on-street parking. However, this extension makes the problem fundamentally harder. Although some research focused on MoD with walking exists (Fielbaum, 2022; Fielbaum et al., 2021), we believe that there is still room for improvement, e.g., using a flexible walking distance, instead of hard constraints (walk towards the vehicle to prevent waiting even if it exceeds maximum walking distance) or exploring the possibility to solve the problem optimally. Also, with the extensively tested simulation stack, we can perform various case studies related to MoD systems. These can be focused, for example, on the impact of large-scale MoD systems on other modes of transport, evaluating different MoD operation policies, the impact of various urban planning decisions on the MoD systems, or the analysis of the trade-offs between capital cost, operational cost, and passenger comfort.

[1]https://movement.uber.com/

Finally, there is an opportunity to further research the optimization in MoD systems. The optimal chaining method we have developed has been so far evaluated only for the dispatching (DARP). However, it can be used for other problems, such as fleet sizing. An evaluation study that would compare the optimal chaining method with other fleet sizing methods, such as the one proposed by Vazifeh et al. (2018) could quantify the benefits of considering users' time constraints in the fleet sizing problem. Also, there is a big challenge in the optimization of the single-vehicle dial-a-ride problem (SVDARP). This problem studied for more than forty years (Desrosiers et al., 1986; Psaraftis, 1980), is now a subproblem of many DARP methods, including the Vehicle-group assignment (VGA) method described in Section 2.3.2. However, the SVDARP is NP-hard and therefore, it is a usually bottleneck of the methods that use it. A typical implementation of the SVDARP is either a dynamic programming algorithm with exponential time complexity (See Desrosiers et al., 1986) or a heuristic that makes the whole method suboptimal (Alonso-Mora et al., 2017). Therefore, any improvement that reduces the time complexity of the SVDARP solution (even if it is still exponential) or a new heuristic that would be better than the current ones would be a big contribution to the field. Among the possible candidates for improvement here are parallelization of the dynamic programming algorithm, implementing of the latest operational research techniques (see e.g. review by Costa et al. (2019)), or using machine learning as a heuristic instead of traditional heuristics like the Insertion Heuristic. The last idea for future research in MoD optimization is to compare the optimal DARP methods on the MoD instances presented in this thesis. So far, we only evaluated the VGA method, as it has a straightforward implementation. However, a huge amount of research exists in the operational research field (Costa et al., 2019; Desaulniers et al., 2020; Gschwind & Irnich, 2014; Ropke & Cordeau, 2009; Sadykov et al., 2021). These methods were so far tested only on classical DARP instances, but they can be potentially less computationally demanding than the VGA method.

APPENDIX A

Publications

This section lists the publications of the author of this thesis. They are divided into two main categories: publications related to the thesis and other publications. Inside each category, the publications are sorted according to the dean's directive no. `FEL_SD_2024_05_V01` into two categories: a) *Journal articles* for the impacted journals indexed in the Web of Science (WoS) and b) *Other WoS publications* for other publications that are indexed in the WoS. The rest of the categories from the directive are skipped, as they are not relevant to the author's publications.

If a publication has been cited, the citation count follows in the format (citations: *WoS IF*, *WoS*, *other*), where:

- *WoS IF* is the number of citations from journals indexed in the WoS with impact factor,
- *WoS* is the number of citations from other publications indexed in the WoS, and
- *other* is the number of citations from other sources.

For contributions, we use the CRediT author statement proposed by Brand et al. (2015). The author statement categories are reproduced in A.1.

Table A.1: CRediT contribution cathegories (Brand et al., 2015).

1	**Conceptualization**	8	**Data curation**
2	Methodology	9	Writing – original draft
3	Software	10	Writing – review and editing
4	Validation	11	Visualization
5	Formal analysis	12	Supervision
6	Investigation	13	Project administration
7	Resources	14	Funding acquisition

A.1 Publications Related to the Thesis

Below, we list the publications that are an integral part of this thesis. To quickly find how each publication relates to the thesis, see Section 1.1.

A.1.1 Journal Articles

(Fiedler, Čertický, Alonso-Mora, et al., 2022) Fiedler, D., Čertický, M., Alonso-Mora, J., Pěchouček, M., & Čáp, M. (2022). Large-scale online ridesharing: The effect of assignment optimality on system performance. *Journal of Intelligent Transportation Systems*, *0*(0), 1–22. https://doi.org/10.1080/15472450.2022.2121651 (contributions: `DF: 1,2,3,4,5,6,8,9,10,11,13; MČe: 2,3,8,9; JAM: 12; MP: 12,14; MČa: 1,2,3,5,10,11,12,13`)

A.1.2 Other WoS Publications

(Fiedler et al., 2017) Fiedler, D., Čáp, M., & Čertický, M. (2017). Impact of mobility-on-demand on traffic congestion: Simulation-based study. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC.2017.8317830 **(citations: 12, 1, 5)** (contributions: `DF: 1,2,3,4,5,6,8,9,10,11; MČe: 2,3,8,9; MČa: 1,2,3,5,9,10,11,12,13,14`)

(Fiedler et al., 2018) Fiedler, D., Čertický, M., Alonso-Mora, J., & Čáp, M. (2018). The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1173–1178. https://doi.org/10.1109/ITSC.2018.8569451 **(citations: 14, 2, 9)** (contributions: `DF: 1,2,3,4,5,6,8,9,10,11,13; MČe: 2,3,8,9; JAM: 12,14; MČa: 1,2,3,5,10,11,12,14`)

(Fiedler & Mrkos, 2023b) Fiedler, D., & Mrkos, J. (2023b). Large-scale Ridesharing DARP Instances Based on Real Travel Demand, 2750–2757. https://doi.org/10.1109/ITSC57777.2023.10422146 (contributions: `DF: 1,2,3,4,5,6,8,9,10,11,12,13; JM: 2,3,4,5,6,10,11`)

A.2 Other Publications

A.2.1 Other WoS Publications

(Fiedler, Čáp, et al., 2022) Fiedler, D., Čáp, M., Nykl, J., & Žilecký, P. (2022). Map Matching Algorithm for Large-scale Datasets. *Proceedings of the 14th International Conference on Agents and Artificial Intelligence Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, 500–508. Retrieved February 24, 2024, from https://www.scitepress.org/Link.aspx?doi=10.5220/0010849100003116 **(citations: 3, 0, 3)** (contributions: `DF: 1,2,3,4,5,6,8,9,10,11,13; MČa: 1,5,10,11,12; JN: 1,2; PŽ: 1,2`)

(Schaefer et al., 2021) Schaefer, M., Čáp, M., Fiedler, D., & Vokřínek, J. (2021). On-demand Robotic Fleet Routing in Capacitated Networks with Time-varying Transportation Demand. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, 907–915. Retrieved February 24, 2024, from https://www.scitepress.org/Link.aspx?doi=10.5220/0010261009070915 (contributions: `DF: 10; MČa: 1,2,5,6,9,10,12; MS: 1,2,3,4,5,6,8,9,10,11,13; JV: 12,14`)

Bibliography

Aditya Ambadipudi, Kersten Heinek, Philipp Kampsho, & Emily Shao. (2017, October 4). *Gauging the disruptive power of robo-taxis in autonomous driving*. McKinsey & Company. Retrieved February 26, 2024, from https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/gauging-the-disruptive-power-of-robo-taxis-in-autonomous-driving?cid=other-eml-alt-mip-mck-oth-1710&hlkid=57ff23622f684cf692d61f51fabf1960&hctky=1320192&hdpid=fb1b0a28-a070-4013-b2b8-91d421cb3623#0

Agatz, N. A. H., Erera, A. L., Savelsbergh, M. W. P., & Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, *45*(9), 1450–1464. https://doi.org/10.1016/j.trb.2011.05.017

Aggregated Reports - TLC. (n.d.). Retrieved February 26, 2024, from https://www.nyc.gov/site/tlc/about/aggregated-reports.page

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993, February). *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Inc.

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

Annie McDonough. (2019, June 13). *New York isn't letting up on ride-hail*. City & State NY. Retrieved February 26, 2024, from https://www.cityandstateny.com/policy/2019/06/new-york-isnt-letting-up-on-ride-hail/177262/

ATOCKAR. (2014, September 15). *Riding with the Stars: Passenger Privacy in the NYC Taxicab Dataset*. Retrieved January 25, 2024, from https://agkn.wordpress.com/2014/09/15/riding-with-the-stars-passenger-privacy-in-the-nyc-taxicab-dataset/

Auld, J., Hope, M., Ley, H., Sokolov, V., Xu, B., & Zhang, K. (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*, 101–116. https://doi.org/10.1016/j.trc.2015.07.017

Auld, J., & Mohammadian, A. (2012). Activity planning processes in the Agent-based Dynamic Activity Planning and Travel Scheduling (ADAPTS) model. *Transportation Research Part A: Policy and Practice*, *46*(8), 1386–1403. https://doi.org/10.1016/j.tra.2012.05.017

Badu-Marfo, G., Farooq, B., & Patterson, Z. (2022). Composite Travel Generative Adversarial Networks for Tabular and Sequential Population Synthesis. *IEEE Transactions* *on Intelligent Transportation Systems*, *23*(10), 17976–17985. https://doi.org/10.1109/TITS.2022.3168232

Balac, M., Becker, H., Ciari, F., & Axhausen, K. W. (2019). Modeling competing free-floating carsharing operators – A case study for Zurich, Switzerland. *Transportation Research Part C: Emerging Technologies*, *98*, 101–117. https://doi.org/10.1016/j.trc.2018.11.011

Banning-Lover, R. (2021). Is ride-hailing finally a profitable business? [newspaper]. *Financial Times*. Retrieved November 26, 2021, from https://www.ft.com/content/bcd71e2a-9ff2-4477-b995-ff82c5530bfb

Bassolas, A., Ramasco, J. J., Herranz, R., & Cantú-Ros, O. G. (2019). Mobile phone records to feed activity-based travel demand models: MATSim for studying a cordon toll policy in Barcelona. *Transportation Research Part A: Policy and Practice*, *121*, 56–74. https://doi.org/10.1016/j.tra.2018.12.024

Beiker, S. (2016). Implementation of an Automated Mobility-on-Demand System. In M. Maurer, J. C. Gerdes, B. Lenz, & H. Winner (Eds.), *Autonomous Driving: Technical, Legal and Social Aspects* (pp. 277–295). Springer. https://doi.org/10.1007/978-3-662-48847-8_14

Beirigo, B. A., Negenborn, R. R., Alonso-Mora, J., & Schulte, F. (2022). A business class for autonomous mobility-on-demand: Modeling service quality contracts in dynamic ridesharing systems. *Transportation Research Part C: Emerging Technologies*, *136*, 103520. https://doi.org/10.1016/j.trc.2021.103520

Ben-Akivai, M., Bowman, J. L., & Gopinath, D. (1996). Travel demand model system for the information era. *Transportation*, *23*(3), 241–266. https://doi.org/10.1007/BF00165704

Beojone, C. V., & Geroliminis, N. (2021). On the inefficiency of ride-sourcing services towards urban congestion. *Transportation Research Part C: Emerging Technologies*, *124*, 102890. https://doi.org/10.1016/j.trc.2020.102890

Berke, A., Doorley, R., Larson, K., & Moro, E. (2022). Generating synthetic mobility data for a realistic population with RNNs to improve utility and privacy. *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 964–967. https://doi.org/10.1145/3477314.3507230

Bhagat, R. (2016, August 25). *World's First Self-Driving Taxis Hit Singapore*. Forbes. Retrieved February 26, 2024, from https://www.forbes.com/sites/rahilbhagat/2016/08/25/worlds-first-self-driving-taxis-hit-singapore-roads/

Bischoff, J., & Maciejewski, M. (2016). Simulation of City-wide Replacement of Private Cars with Autonomous Taxis in Berlin. *Procedia Computer Science*, *83*, 237–244. https://doi.org/10.1016/j.procs.2016.04.121

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

Bob Pishue. (2023, January). *2022 INRIX Global Traffic Scorecard*. INRIX.

Boogaard, H., Patton, A. P., Atkinson, R. W., Brook, J. R., Chang, H. H., Crouse, D. L., Fussell, J. C., Hoek, G., Hoffmann, B., Kappeler, R., Kutlar Joss, M., Ondras, M., Sagiv, S. K., Samoli, E., Shaikh, R., Smargiassi, A., Szpiro, A. A., Van Vliet, E. D. S., Vienneau, D., . . . Forastiere, F. (2022). Long-term exposure to traffic-related air pollution and selected health outcomes: A systematic review and meta-analysis. *Environment International*, *164*, 107262. https://doi.org/10.1016/j.envint.2022.107262

Bösch, P., Kirill, M., & Ciari, F. (2016). The IVT 2015 baseline scenario.

Braekers, K., Caris, A., & Janssens, G. K. (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, *67*, 166–186. https://doi.org/10.1016/j.trb.2014.05.007

Brand, A., Allen, L., Altman, M., Hlava, M., & Scott, J. (2015). Beyond authorship: Attribution, contribution, collaboration, and credit. *Learned Publishing*, *28*(2), 151–155. https://doi.org/10.1087/20150211

Burns, L. D., Jordan, W. C., & Scarborough, B. A. (2013). *Transforming Personal Mobility*. Earth Institute, Columbia University.

Campbell, A., & Savelsbergh, M. (2004). Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science*, *38*, 369–378. https://doi.org/10.1287/trsc.1030.0046

Cano, R. (2023, August 8). *Cruise, Waymo reveal how many driverless cars they have in S.F. and how often they stall on city streets*. San Francisco Chronicle. Retrieved February 26, 2024, from https://www.sfchronicle.com/sf/article/cruise-waymo-driverless-cars-in-s-f-18282902.php

Čáp, M., & Alonso-Mora, J. (2018). Multi-Objective Analysis of Ridesharing in Automated Mobility-on-Demand. *Robotics: Science and Systems XIV*. https://doi.org/10.15607/RSS.2018.XIV.039

Čertický, M., Drchal, J., Cuchý, M., & Jakob, M. (2015). Fully agent-based simulation model of multimodal mobility in European cities. *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 229–236. https://doi.org/10.1109/MTITS.2015.7223261

Choi, J., Kim, Y., Kwak, M., Park, M., & Lee, D. (2023). Measuring taxi ridesharing effects and its spatiotemporal pattern in Seoul, Korea. *Travel Behaviour and Society*, *30*, 148–162. https://doi.org/10.1016/j.tbs.2022.09.001

Cookson, G. (2018). INRIX Global Traffic Scorecard, 44.

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

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

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

Costa, L., Contardo, C., & Desaulniers, G. (2019). Exact Branch-Price-and-Cut Algorithms for Vehicle Routing. *Transportation Science*, *53*(4), 946–985. https://doi.org/10.1287/trsc.2018.0878

Culnane, D. C., Rubinstein, A. B. I. P., & Teague, A. V. (2019, August 14). *Stop the Open Data Bus, We Want to Get Off*. arXiv: 1908.05004 [cs]. https://doi.org/10.48550/arXiv.1908.05004

Davis, A. Y., Pijanowski, B. C., Robinson, K. D., & Kidwell, P. B. (2010). Estimating parking lot footprints in the Upper Great Lakes Region of the USA. *Landscape and Urban Planning*, *96*(2), 68–77. https://doi.org/10.1016/j.landurbplan.2010.02.004

Desaulniers, G., Gschwind, T., & Irnich, S. (2020). Variable Fixing for Two-Arc Sequences in Branch-Price-and-Cut Algorithms on Path-Based Models. *Transportation Science*, *54*(5), 1170–1188. https://doi.org/10.1287/trsc.2020.0988

Desrosiers, J., Dumas, Y., & Soumis, F. (1986). A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-A-Ride Problem with Time Windows. *American Journal of Mathematical and Management Sciences*, *6*(3-4), 301–325. https://doi.org/10.1080/01966324.1986.10737198

Drchal, J., Čertický, M., & Jakob, M. (2016). VALFRAM: Validation Framework for Activity-Based Models. *Journal of Artificial Societies and Social Simulation*, *19*(3), 5.

Drchal, J., Čertický, M., & Jakob, M. (2019). Data-driven activity scheduler for agent-based mobility models. *Transportation Research Part C: Emerging Technologies*, *98*, 370–390. https://doi.org/10.1016/j.trc.2018.12.002

Engelhardt, R., Dandl, F., Bilali, A., & Bogenberger, K. (2019). Quantifying the Benefits of Autonomous On-Demand Ride-Pooling: A Simulation Study for Munich, Germany. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2992–2997. https://doi.org/10.1109/ITSC.2019.8916955

Fagnant, D., & Kockelman, K. (2014). The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies*, *40*, 1–13. https://doi.org/10.1016/j.trc.2013.12.001

Febbraro, A. D., Giglio, D., & Sacco, N. (2018). On Exploiting Ride-Sharing and Crowd-Shipping Schemes within the Physical Internet Framework. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1493–1500. https://doi.org/10.1109/ITSC.2018.8569761

Feng, S., Duan, P., Ke, J., & Yang, H. (2022). Coordinating ride-sourcing and public transport services with a reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, *138*, 103611. https://doi.org/10.1016/j.trc.2022.103611

Ferguson, E. (1997). The rise and fall of the American carpool: 1970–1990. *Transportation*, *24*(4), 349–376. https://doi.org/10.1023/A:1004928012320

Fiedler, D., Čáp, M., & Čertický, M. (2017). Impact of mobility-on-demand on traffic congestion: Simulation-based study. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC.2017.8317830

Fiedler, D., Čáp, M., Nykl, J., & Žilecký, P. (2022). Map Matching Algorithm for Large-scale Datasets. *Proceedings of the 14th International Conference on Agents and Artificial Intelligence Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, 500–508. Retrieved February 24, 2024, from https://www.scitepress.org/Link.aspx?doi=10.5220/0010849100003116

Fiedler, D., Čertický, M., Alonso-Mora, J., & Čáp, M. (2018). The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1173–1178. https://doi.org/10.1109/ITSC.2018.8569451

Fiedler, D., Čertický, M., Alonso-Mora, J., Pěchouček, M., & Čáp, M. (2022). Large-scale online ridesharing: The effect of assignment optimality on system performance. *Journal of Intelligent Transportation Systems*, *0*(0), 1–22. https://doi.org/10.1080/15472450.2022.2121651

Fiedler, D., Difonzo, F. V., & Mrkos, J. (2024, February 24). *Optimal Chaining of Vehicle Plans with Time Windows*. arXiv: 2401.02873 `[cs, math]`. https://doi.org/10.48550/arXiv.2401.02873

Fiedler, D., & Mrkos, J. (2023a, May 30). *Large-scale Ridesharing DARP Instances Based on Real Travel Demand*. arXiv: 2305.18859 `[cs, math]`. https://doi.org/10.48550/arXiv.2305.18859

Fiedler, D., & Mrkos, J. (2023b). Large-scale Ridesharing DARP Instances Based on Real Travel Demand, 2750–2757. https://doi.org/10.1109/ITSC57777.2023.10422146

Fielbaum, A. (2022). Optimizing a vehicle's route in an on-demand ridesharing system in which users might walk. *Journal of Intelligent Transportation Systems*, *26*(4), 432–447. https://doi.org/10.1080/15472450.2021.1901225

Fielbaum, A., Bai, X., & Alonso-Mora, J. (2021). On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C: Emerging Technologies*, *126*, 103061. https://doi.org/10.1016/j.trc.2021.103061

Fiore, M., Katsikouli, P., Zavou, E., Cunche, M., Fessant, F., Hello, D. L., Aivodji, U. M., Olivier, B., Quertier, T., & Stanica, R. (2020). Privacy in trajectory micro-data publishing: A survey. *Transactions on Data Privacy*, *13*, 91. Retrieved January 25, 2024, from https://inria.hal.science/hal-02968279

Gambs, S., Killijian, M.-O., & Núñez del Prado Cortez, M. (2014). De-anonymization attack on geolocated data. *Journal of Computer and System Sciences*, *80*(8), 1597–1614. https://doi.org/10.1016/j.jcss.2014.04.024

Godart, A., Manier, H., Bloch, C., & Manier, M.-A. (2018). MILP for a Variant of Pickup Delivery Problem for both Passengers and Goods Transportation. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2692–2698. https://doi.org/10.1109/SMC.2018.00460

Gower, P., & Spicer, D. (1932). Aviation as a Career. *The Woman Engineer*, *3*(111), 151.

Gschwind, T., & Drexl, M. (2019). Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem. *Transportation Science*. https://doi.org/10.1287/trsc.2018.0837

Gschwind, T., & Irnich, S. (2014). Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem. *Transportation Science*, *49*(2), 335–354. https://doi.org/10.1287/trsc.2014.0531

Guenther, K. W., & Givens, W. E. (1970, February 1). *The Courier: A Prototype Vehicle For Dial-A-Ride Service* (SAE Technical Paper No. 700186). SAE International. Warrendale, PA. https://doi.org/10.4271/700186

Guo, Z., Hao, M., Yu, B., & Yao, B. (2021). Robust minimum fleet problem for autonomous and human-driven vehicles in on-demand ride services considering mixed operation zones. *Transportation Research Part C: Emerging Technologies*, *132*, 103390. https://doi.org/10.1016/j.trc.2021.103390

Gurobi Optimization, LLC. (2023). Gurobi optimizer reference manual. https://www.gurobi.com

Haliem, M., Mani, G., Aggarwal, V., & Bhargava, B. (2021). A Distributed Model-Free Ride-Sharing Approach for Joint Matching, Pricing, and Dispatching Using Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems*, *22*(12), 7931–7942. https://doi.org/10.1109/TITS.2021.3096537

Ham, A. (2021). Dial-a-ride problem: Mixed integer programming revisited and constraint programming proposed. *Engineering Optimization*, *0*(0), 1–14. https://doi.org/10.1080/0305215X.2021.1996570

Hartgen, D. T. (1977). Ridesharing Behavior: A Review of Recent Findings. (Prelim. Res Rpt. 130). Retrieved June 5, 2020, from https://trid.trb.org/view/68431

Hensher, D. A. (2007, September 14). *Handbook of Transport Modelling* (K. J. Button, Ed.; 2nd edition). Emerald Publishing.

Ho, S. C., Szeto, W. Y., Kuo, Y.-H., Leung, J. M. Y., Petering, M., & Tou, T. W. H. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, *111*, 395–421. https://doi.org/10.1016/j.trb.2018.02.001

Hörl, S., Ruch, C., Becker, F., Frazzoli, E., & Axhausen, K. W. (2019). Fleet operational policies for automated mobility: A simulation assessment for Zurich. *Transportation Research Part C: Emerging Technologies*, *102*, 20–31. https://doi.org/10.1016/j.trc.2019.02.020

Hyland, M., & Mahmassani, H. S. (2018). Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests. *Transportation Research Part C: Emerging Technologies*, *92*, 278–297. https://doi.org/10.1016/j.trc.2018.05.003

INRIX. (2017, June 17). *Searching for Parking Costs Americans $73 Billion a Year*. Inrix. Retrieved February 26, 2024, from https://inrix.com/press-releases/parking-pain-us/

International Transport Forum. (2024). *Transport infrastructure investment and maintenance*. https://doi.org/https://doi.org/10.1787/g2g55573-en

Jakob, M., Moler, Z., Komenda, A., Yin, Z., Jiang, A. X., Johnson, M. P., Pěchouček, M., & Tambe, M. (2012). AgentPolis: Towards a platform for fully agent-based modeling of multi-modal transportation (demonstration). *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, *3*, 1501–1502.

Jaw, J.-J., Odoni, A. R., Psaraftis, H. N., & Wilson, N. H. M. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, *20*(3), 243–257. https://doi.org/10.1016/0191-2615(86)90020-2

Jung, J., Jayakrishnan, R., & Young Park, J. (2015). Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing. *Computer-Aided Civil and Infrastructure Engineering*, *31*(4), 275–291. https://doi.org/10.1111/mice.12157

Kalina, P., Vokřínek, J., & Mařík, V. (2015). Agents Toward Vehicle Routing Problem With Time Windows. *Journal of Intelligent Transportation Systems*, *19*(1), 3–17. https://doi.org/10.1080/15472450.2014.889953

Kapp, A., Hansmeyer, J., & Mihaljević, H. (2023). Generative Models for Synthetic Urban Mobility Data: A Systematic Literature Review. *ACM Computing Surveys*, *56*(4), 93:1–93:37. https://doi.org/10.1145/3610224

Karp, R. M. (1972). Reducibility among Combinatorial Problems. In R. E. Miller, J. W. Thatcher, & J. D. Bohlinger (Eds.), *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department* (pp. 85–103). Springer US. https://doi.org/10.1007/978-1-4684-2001-2_9

Kirchler, D., & Wolfler Calvo, R. (2013). A Granular Tabu Search algorithm for the Dial-a-Ride Problem. *Transportation Research Part B: Methodological*, *56*, 120–135. https://doi.org/10.1016/j.trb.2013.07.014

Komenda, A., Vokřínek, J., Čáp, M., & Pěchouček, M. (2013). Developing Multiagent Algorithms for Tactical Missions Using Simulation. *IEEE Intelligent Systems*, *28*(1), 42–49. https://doi.org/10.1109/MIS.2012.90

Kondor, D., Bojic, I., Resta, G., Duarte, F., Santi, P., & Ratti, C. (2022). The cost of non-coordination in urban on-demand mobility. *Scientific Reports*, *12*(1), 4669. https://doi.org/10.1038/s41598-022-08427-2

Kumar, R. R., Varakantham, P., & Cheng, S.-F. (2023). Strategic Planning for Flexible Agent Availability in Large Taxi Fleets. *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 552–560.

Lau, S. T., & Susilawati, S. (2021). Shared autonomous vehicles implementation for the first and last-mile services. *Transportation Research Interdisciplinary Perspectives*, *11*, 100440. https://doi.org/10.1016/j.trip.2021.100440

Li, J., Embry, P., Mattingly, S. P., Sadabadi, K. F., Rasmidatta, I., & Burris, M. W. (2007). Who Chooses to Carpool and Why?: Examination of Texas Carpoolers. *Transportation Research Record*. https://doi.org/10.3141/2021-13

Li, M., Di, X., Liu, H. X., & Huang, H.-J. (2020). A restricted path-based ridesharing user equilibrium. *Journal of Intelligent Transportation Systems*, *24*(4), 383–403. https://doi.org/10.1080/15472450.2019.1658525

Lokhandwala, M., & Cai, H. (2018). Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of NYC. *Transportation Research Part C: Emerging Technologies*, *97*, 45–60. https://doi.org/10.1016/j.trc.2018.10.007

Los, J., Schulte, F., Gansterer, M., Hartl, R. F., Spaan, M. T. J., & Negenborn, R. R. (2022). Large-scale collaborative vehicle routing. *Annals of Operations Research*. https://doi.org/10.1007/s10479-021-04504-3

Lotfi, S., & Abdelghany, K. (2022). Ride matching and vehicle routing for on-demand mobility services. *Journal of Heuristics*, *28*(3), 235–258. https://doi.org/10.1007/s10732-022-09491-7

Lu, Y., Basak, K., Carrion, C., Loganathan, H., Adnan, M., Pereira, F., Saber, V. H., & Ben-Akiva, M. (2015). SimMobility Mid-Term Simulator: A State of the Art Integrated Agent Based Demand and Supply Model. *94th Annual Meeting of Transportation Research Board*. Retrieved January 27, 2024, from https://www.semanticscholar.org/paper/SimMobility-Mid-Term-Simulator%3A-A-State-of-the-Art-Lu-Basak/687c4e86780d4834a15b72f293c055b771388572

Lyu, Z., & Yu, A. J. (2023). The pickup and delivery problem with transshipments: Critical review of two existing models and a new formulation. *European Journal of Operational Research*, *305*(1), 260–270. https://doi.org/10.1016/j.ejor.2022.05.053

Ma, J., Xu, M., Meng, Q., & Cheng, L. (2020). Ridesharing user equilibrium problem under OD-based surge pricing strategy. *Transportation Research Part B: Methodological*, *134*, 1–24. https://doi.org/10.1016/j.trb.2020.02.001

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

Ma, T.-Y., Rasulkhani, S., Chow, J. Y. J., & Klein, S. (2019). A dynamic ridesharing dispatch and idle vehicle repositioning strategy with integrated transit transfers. *Transportation Research Part E: Logistics and Transportation Review*, *128*, 417–442. https://doi.org/10.1016/j.tre.2019.07.002

Maciejewski, M., & Bischoff, J. (2018). Congestion effects of autonomous taxi fleets. *Transport*, *33*(4), 971–980. https://doi.org/10.3846/16484142.2017.1347827

Manchella, K., Umrawal, A. K., & Aggarwal, V. (2021). FlexPool: A Distributed Model-Free Deep Reinforcement Learning Algorithm for Joint Passengers and Goods Transportation. *IEEE Transactions on Intelligent Transportation Systems*, *22*(4), 2035–2047. https://doi.org/10.1109/TITS.2020.3048361

Marczuk, K. A., Hong, H. S. S., Azevedo, C. M. L., Adnan, M., Pendleton, S. D., Frazzoli, E., & Lee, D. H. (2015). Autonomous mobility on demand in SimMobility: Case study of the central business district in Singapore. *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 167–172. https://doi.org/10.1109/ICCIS.2015.7274567

Masmoudi, M. A., Hosny, M., Braekers, K., & Dammak, A. (2016). Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, *96*, 60–80. https://doi.org/10.1016/j.tre.2016.10.002

Masmoudi, M. A., Hosny, M., Demir, E., & Pesch, E. (2020). Hybrid adaptive large neighborhood search algorithm for the mixed fleet heterogeneous dial-a-ride problem. *Journal of Heuristics*, *26*(1), 83–118. https://doi.org/10.1007/s10732-019-09424-x

Miller, J., & How, J. P. (2017). Predictive positioning and quality of service ridesharing for campus mobility on demand systems. *2017 IEEE International Conference on*

Robotics and Automation (ICRA), 1402–1408. https://doi.org/10.1109/ICRA.2017.7989167

Mitchell, W. J., Hainley, B. E., & Burns, L. D. (2010, January 29). *Reinventing the Automobile: Personal Urban Mobility for the 21st Century*. The MIT Press. https://doi.org/10.7551/mitpress/8490.001.0001

Muelas, S., LaTorre, A., & Peña, J.-M. (2013). A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Systems with Applications*, *40*(14), 5516–5531. https://doi.org/10.1016/j.eswa.2013.04.015

Muelas, S., LaTorre, A., & Peña, J.-M. (2015). A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, *54*, 110–130. https://doi.org/10.1016/j.trc.2015.02.024

Nair, R., Miller-Hooks, E., Hampshire, R. C., & Bušić, A. (2013). Large-Scale Vehicle Sharing Systems: Analysis of Vélib'. *International Journal of Sustainable Transportation*, *7*(1), 85–106. https://doi.org/10.1080/15568318.2012.660115

Najmi, A., Rey, D., & Rashidi, T. H. (2017). Novel dynamic formulations for real-time ride-sharing systems. *Transportation Research Part E: Logistics and Transportation Review*, *108*, 122–140. https://doi.org/10.1016/j.tre.2017.10.009

National Research Council (U S.), Highway Research Board. (1971). *Demand-Actuated Transportation Systems*. The Board.

Neumann, A., Balmer, M., & Rieser, M. (2012). Converting a Static Macroscopic Model Into a Dynamic Activity-Based Model for Analyzing Public Transport Demand in Berlin. *Proceedings of the 13th Conference of the International Association for Travel Behaviour Research (IATBR)*.

NYC Department of Transportation. (2019). *New York City Mobility Report*. NYC Department of Transportation. Retrieved November 7, 2023, from https://www.nyc.gov/html/dot/downloads/pdf/mobility-report-2019-print.pdf

NYC Taxi & Limousine Commission. (2018). *2018 Factbook*. NYC Taxi & Limousine Commission.

Oke, J. B., Akkinepally, A. P., Chen, S., Xie, Y., Aboutaleb, Y. M., Azevedo, C. L., Zegras, P. C., Ferreira, J., & Ben-Akiva, M. (2020). Evaluating the systemic effects of automated mobility-on-demand services via large-scale agent-based simulation of auto-dependent prototype cities. *Transportation Research Part A: Policy and Practice*, *140*, 98–126. https://doi.org/10.1016/j.tra.2020.06.013

Ouyang, K., Shokri, R., Rosenblum, D. S., & Yang, W. (2018). A Non-Parametric Generative Model for Human Trajectories, 3812–3817. Retrieved January 25, 2024, from https://www.ijcai.org/proceedings/2018/530

Pang, Y., Tsubouchi, K., Yabe, T., & Sekimoto, Y. (2017). Modeling and reproducing human daily travel behavior from GPS data: A Markov Decision Process approach. *Proceedings of the 1st ACM SIGSPATIAL Workshop on Prediction of Human Mobility*, 1–9. https://doi.org/10.1145/3152341.3152347

Parragh, S. N. (2011). Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, *19*(5), 912–930. https://doi.org/10.1016/j.trc.2010.06.002

Pavone, M., Smith, S. L., Frazzoli, E., & Rus, D. (2012). Robotic load balancing for mobility-on-demand systems. *The International Journal of Robotics Research*, *31*(7), 839–854. https://doi.org/10.1177/0278364912444766

Pfeiffer, C., & Schulz, A. (2022). An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization. *OR Spectrum*, *44*(1), 87–119. https://doi.org/10.1007/s00291-021-00656-7

Pierotti, J., & van Essen, J. T. (2021). MILP models for the Dial-a-Ride problem with transfers. *EURO Journal on Transportation and Logistics*, *10*(100037). https://doi.org/10.1016/j.ejtl.2021.100037

Psaraftis, H. N. (1980). A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, *14*(2), 130–154. https://doi.org/10.1287/trsc.14.2.130

Qu, B., Mao, L., Xu, Z., Feng, J., & Wang, X. (2022). How Many Vehicles Do We Need? Fleet Sizing for Shared Autonomous Vehicles With Ridesharing. *IEEE Transactions on Intelligent Transportation Systems*, *23*(9), 14594–14607. https://doi.org/10.1109/TITS.2021.3130749

Resolution Approving Authorization for Cruise Llc's Expanded Service in Autonomous Vehicle Passenger Service Phase I Driverless Deployment Program (Resolution No. TL-19145). (2023a, August 10). Public Utilities Commission of the State of California. Retrieved February 26, 2024, from https://docs.cpuc.ca.gov/PublishedDocs/Published/G000/M516/K812/516812218.PDF

Resolution Approving Waymo Llc's Application for Phase I Driverless Autonomous Vehicle Passenger Service Deployment Program (Resolution No. TL-19144). (2023b, August 10). Public Utilities Commission of the State of California. Retrieved February 26, 2024, from https://docs.cpuc.ca.gov/PublishedDocs/Published/G000/M516/K812/516812344.PDF

Road Transport Year Book. (2023, April 13). Goverment of India, Ministry of Road Transportat & Highways. New Delhi.

Ropke, S., & Cordeau, J.-F. (2009). Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*. https://doi.org/10.1287/trsc.1090.0272

Ropke, S., Cordeau, J.-F., & Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, *49*(4), 258–272. https://doi.org/10.1002/net.20177

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, *40*(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

Sadykov, R., Uchoa, E., & Pessoa, A. (2021). A Bucket Graph–Based Labeling Algorithm with Application to Vehicle Routing. *Transportation Science*, *55*(1), 4–28. https://doi.org/10.1287/trsc.2020.0985

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

Santos, D. O., & Xavier, E. C. (2013). Dynamic Taxi and Ridesharing: A Framework and Heuristics for the Optimization Problem. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2885–2891. Retrieved March 28, 2019, from http://dl.acm.org/citation.cfm?id=2540128.2540544

Santos, D. O., & Xavier, E. C. (2015). Taxi and Ride Sharing: A Dynamic Dial-a-Ride Problem with Money as an Incentive. *Expert Systems with Applications*, *42*(19), 6728–6737. https://doi.org/10.1016/j.eswa.2015.04.060

Savelsbergh, M. W. P. (1985). Local search in routing problems with time windows. *Annals of Operations Research*, *4*(1), 285–305. https://doi.org/10.1007/BF02022044

Schaefer, M., Čáp, M., Fiedler, D., & Vokřínek, J. (2021). On-demand Robotic Fleet Routing in Capacitated Networks with Time-varying Transportation Demand. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, 907–915. Retrieved February 24, 2024, from https://www.scitepress.org/Link.aspx?doi=10.5220/0010261009070915

Seo, T., & Asakura, Y. (2022). Multi-Objective Linear Optimization Problem for Strategic Planning of Shared Autonomous Vehicle Operation and Infrastructure Design. *IEEE Transactions on Intelligent Transportation Systems*, *23*(4), 3816–3828. https://doi.org/10.1109/TITS.2021.3071512

Shaheen, S. A., & Cohen, A. P. (2013). Carsharing and Personal Vehicle Services: Worldwide Market Developments and Emerging Trends. *International Journal of Sustainable Transportation*, *7*(1), 5–34. https://doi.org/10.1080/15568318.2012.660103

Shoup, D. (2014). The high cost of minimum parking requirements. *Transport and Sustainability*, *5*, 87–113. https://doi.org/10.1108/S2044-994120140000005011

Sir Walter Gilbey. (1903). *Early Carriages and Roads*. Vinton & co., 1td. Retrieved February 26, 2024, from http://archive.org/details/earlycarriagesa00gilbgoog

Spieser, K., Treleaven, K., Zhang, R., Frazzoli, E., Morton, D., & Pavone, M. (2014). Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore. In G. Meyer & S. Beiker (Eds.), *Road Vehicle Automation* (pp. 229–245). Springer International Publishing. https://doi.org/10.1007/978-3-319-05990-7_20

Srivatsa, M., & Hicks, M. (2012). Deanonymizing mobility traces: Using social network as a side-channel. *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 628–637. https://doi.org/10.1145/2382196.2382262

Susan Shaheen & Adam Cohen. (2020). Similarities and Differences of Mobility on Demand (MOD) and Mobility as a Service (MaaS) — Transportation Sustainability Research Center. *ite journal*, *90*(6), 29–35. Retrieved September 9, 2021, from https://tsrc.berkeley.edu/publications/similarities-and-differences-mobility-demand-mod-and-mobility-service-maas

Tachet, R., Sagarra, O., Santi, P., Resta, G., Szell, M., Strogatz, S. H., & Ratti, C. (2017). Scaling Law of Urban Ride Sharing. *Scientific Reports*, *7*, 42868. https://doi.org/10.1038/srep42868

Tadaki, S.-i., Kikuchi, M., Fukui, M., Nakayama, A., Nishinari, K., Shibata, A., Sugiyama, Y., Yosida, T., & Yukawa, S. (2015). Critical Density of Experimental Traffic Jam. In M. Chraibi, M. Boltes, A. Schadschneider, & A. Seyfried (Eds.), *Traffic and Granular*

Flow '13 (pp. 505–511). Springer International Publishing. https://doi.org/10.1007/978-3-319-10629-8_56

Tajaddini, A., Rose, G., M. Kockelman, K., & L. Vu, H. (2021). Recent Progress in Activity-Based Travel Demand Modeling: Rising Data and Applicability (S. De Luca, R. Di Pace, & C. Fiori, Eds.). *Models and Technologies for Smart, Sustainable and Safe Transportation Systems*. https://doi.org/10.5772/intechopen.93827

Thangaraj, R. S., Mukherjee, K., Raravi, G., Metrewar, A., Annamaneni, N., & Chattopadhyay, K. (2017). Xhare-a-Ride: A Search Optimized Dynamic Ride Sharing System with Approximation Guarantee. *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 1117–1128. https://doi.org/10.1109/ICDE.2017.156

Toole, J. L., Colak, S., Sturt, B., Alexander, L. P., Evsukoff, A., & González, M. C. (2015). The path most traveled: Travel demand estimation using big data resources. *Transportation Research Part C: Emerging Technologies*, *58*, 162–177. https://doi.org/10.1016/j.trc.2015.04.022

Toth, P., & Vigo, D. (2014, December 5). *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM.

TU, M., LI, Y., LI, W., TU, M., Orfila, O., Gruyer, D., & BAN, X. (2019). Improving Ridesplitting Service Using Optimization Procedures on Shareability Network: A Case Study of Chengdu, China. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 4506–4511. https://doi.org/10.1109/ITSC.2019.8917402

Tuncel, K., Koutsopoulos, H. N., & Ma, Z. (2023). An integrated ride-matching and vehicle-rebalancing model for shared mobility on-demand services. *Computers & Operations Research*, *159*, 106317. https://doi.org/10.1016/j.cor.2023.106317

United Nations. (2018, May 16). *68% of the world population projected to live in urban areas by 2050, says UN — UN DESA — United Nations Department of Economic and Social Affairsssss*. Retrieved January 30, 2019, from https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

van Engelen, M., Cats, O., Post, H., & Aardal, K. (2018). Enhancing flexible transport services with demand-anticipatory insertion heuristics. *Transportation Research Part E: Logistics and Transportation Review*, *110*, 110–121. https://doi.org/10.1016/j.tre.2017.12.015

Vazifeh, M. M., Santi, P., Resta, G., Strogatz, S. H., & Ratti, C. (2018). Addressing the minimum fleet problem in on-demand urban mobility. *Nature*, *557*(7706), 534–538. https://doi.org/10.1038/s41586-018-0095-1

Vehicles in Use Europe 2022. (2022, January). European Automobile Manufacturers' Association.

Voigt, S., & Kuhn, H. (2022). Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. *Networks*, *79*(3), 403–426. https://doi.org/10.1002/net.22045

Wallar, A., Alonso-Mora, J., & Rus, D. (2019a). Optimizing Vehicle Distributions and Fleet Sizes for Shared Mobility-on-Demand. *2019 International Conference on Robotics and Automation (ICRA)*, 3853–3859. https://doi.org/10.1109/ICRA.2019.8793685

Wallar, A., Alonso-Mora, J., & Rus, D. (2019b). Optimizing Vehicle Distributions and Fleet Sizes for Shared Mobility-on-Demand. *2019 International Conference on Robotics and Automation (ICRA)*, 3853–3859. https://doi.org/10.1109/ICRA.2019.8793685

Wang, C., Song, Y., Wei, Y., Fan, G., Jin, H., & Zhang, F. (2021). Towards Minimum Fleet for Ridesharing-Aware Mobility-on-Demand Systems. *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488862

Widhalm, P., Yang, Y., Ulm, M., Athavale, S., & González, M. C. (2015). Discovering urban activity patterns in cell phone data. *Transportation*, *42*(4), 597–623. https://doi.org/10.1007/s11116-015-9598-x

Wollenstein-Betech, S., Paschalidis, I. C., & Cassandras, C. G. (2021). Optimal Operations Management of Mobility-on-Demand Systems. *Frontiers in Sustainable Cities*, *3*. Retrieved April 6, 2023, from https://www.frontiersin.org/articles/10.3389/frsc.2021.681096

Xu, Y., Wang, W., Xiong, G., Liu, X., Wu, W., & Liu, K. (2022). Network-Flow-Based Efficient Vehicle Dispatch for City-Scale Ride-Hailing Systems. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 5526–5538. https://doi.org/10.1109/TITS.2021.3054893

Yan, C.-Y., Hu, M.-B., Jiang, R., Long, J., Chen, J.-Y., & Liu, H.-X. (2019). Stochastic Ridesharing User Equilibrium in Transport Networks. *Networks and Spatial Economics*, *19*(4), 1007–1030. https://doi.org/10.1007/s11067-019-9442-5

Yin, M., Sheehan, M., Feygin, S., Paiement, J.-F., & Pozdnoukhov, A. (2018). A Generative Model of Urban Activities from Cellular Data. *IEEE Transactions on Intelligent Transportation Systems*, *19*(6), 1682–1696. https://doi.org/10.1109/TITS.2017.2695438

Yu, B., Ma, Y., Xue, M., Tang, B., Wang, B., Yan, J., & Wei, Y.-M. (2017). Environmental benefits from ridesharing: A case of Beijing. *Applied Energy*, *191*, 141–152. https://doi.org/10.1016/j.apenergy.2017.01.052

Yu, X., Chen, J., Kumar, P., Khani, A., & Mao, H. (2022). An integrated optimisation framework for locating depots in shared autonomous vehicle systems. *Transportmetrica A: Transport Science*, *0*(0), 1–39. https://doi.org/10.1080/23249935.2022.2152299

Zardini, G., Lanzetti, N., Pavone, M., & Frazzoli, E. (2022). Analysis and control of autonomous mobility-on-demand systems. *Annual Review of Control, Robotics, and Autonomous Systems*, *5*(1), 633–658. https://doi.org/10.1146/annurev-control-042920-012811

Zhan, X., Szeto, W. Y., & (Michael) Chen, X. (2022). A simulation–optimization framework for a dynamic electric ride-hailing sharing problem with a novel charging strategy. *Transportation Research Part E: Logistics and Transportation Review*, *159*, 102615. https://doi.org/10.1016/j.tre.2022.102615

Zhan, X., Szeto, W. Y., Shui, C. S., & Chen, X. (2021). A modified artificial bee colony algorithm for the dynamic ride-hailing sharing problem. *Transportation Research Part E: Logistics and Transportation Review*, *150*, 102124. https://doi.org/10.1016/j.tre.2020.102124

Zhang, R., Rossi, F., & Pavone, M. (2016a). Model predictive control of autonomous mobility-on-demand systems. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1382–1389. https://doi.org/10.1109/ICRA.2016.7487272

Zhang, R., Rossi, F., & Pavone, M. (2016b). Routing Autonomous Vehicles in Congested Transportation Networks: Structural Properties and Coordination Algorithms. https://doi.org/10.15607/RSS.2016.XII.032

Zhang, W., Jacquillat, A., Wang, K., & Wang, S. (2023). Routing Optimization with Vehicle–Customer Coordination. *Management Science*. https://doi.org/10.1287/mnsc.2023.4739