universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

"Models and algorithms for shared mobility systems"

verfasst von / submitted by

Miriam Enzi

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien, 2021 / Vienna 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 794 370 403
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Logistics and Operations Management
Betreut von / Supervisor:	Univ.-Prof.in Sophie Parragh, PhD.

Acknowledgements

During the past years as a PhD-candidate I was lucky enough to receive support from many sides. I want to take this opportunity to acknowledge the importance of these people on my way.

First and foremost, I want to thank my supervisor Prof. Sophie N. Parragh for her exceptional support, patience and guidance over the past years. Her extensive input, comprehensive feedback and encouraging words are the foundation of this thesis.

Moreover, I want to thank Dr. Matthias Prandstetter, who gave me the opportunity to work as a PhD-researcher at the AIT and provided initial ideas and feedback. Matthias also connected me with Prof. Jakob Puchinger, who I wish to thank for the support on Chapter 4 and for giving me the opportunity to spend three months at the CentraleSupélec as a visiting researcher.

Furthermore, I want to thank Prof. David Pisinger who I met during my first year as a PhD-candidate. He turned out to be an indispensable support on this thesis. We had fruitful discussions and fun working sessions during various visits and a longer research stay at the DTU.

I want to thank the many peers who I met at the Vienna University of Economics and Business, the Austrian Institute of Technology, the University of Vienna, the Johannes Kepler University in Linz, as well as the CentraleSupélec and the DTU Denmark.

I would like to thank my family, especially my parents, who gave me the support and freedom to go to university, where I found my passion for Operations Research. A big "thank you" also to my friends, who help me during difficult times and celebrate good times with me. Finally, I want to express the deepest gratitude to Dominik for the fundamental support, encouragement, and patience.

Preface

This thesis consists of research that I have conducted over the past years at the AIT Austrian Institute of Technology and the Johannes Kepler University Linz, mainly contributing to two research projects, whilst being a PhD student at the University of Vienna. At the AIT I was working for three years under the project SEAMLESS (Climate and Energy Funds (KliEn), grant number 853767), at the JKU I conducted research for the project MOMIP: Multi-objective (mixed) integer programming (Austrian Science Fund (FWF), P 31366). Parts of the book have already been published at international scientific journals or are still under revision for publication, and have been presented at scientific conferences.

More precisely, Chapter 2 has been submitted to the "EURO Journal on Transportation and Logistics". Chapter 3 with the title "Modeling and solving the multimodal car- and ride-sharing problem" is published at the "European Journal of Operational Research", while Chapter 4 "The bi-objective multimodal car-sharing problem" has been accepted for publication at the "OR Spectrum". At the beginning of the PhD the conference paper "Planning Shared Corporate Mobility Services" has been published in the "Transportation Research Procedia", which, however, is not part of this thesis.

The results of this work have been presented at several international scientific conferences: at the triennial conference of the International Federation of Operational Research Societies (IFORS) in 2017, the Euro Working Group on Transportation (EWGT) in 2017, the Workshop on Young Academics' Management Science (YAMS) in 2017, the joint international conference on Applied Combinatorial Optimization of the European Operational Research Societies (EURO) and the Association of Latin-Iberoamerican Operational Research Societies (ALIO) in 2018, the European Conference on Operational Research by the EURO in 2018 and 2019, the INFORMS Transportation Science and Logistics (TSL) Society Workshop in 2019, and the annual scientific conference of the national Operations Research Societies of Germany in 2019. Moreover, during the course of this thesis two international co-operations with professors were maintained, during which talks in research seminars have been conducted. I gave three talks in research seminars at the Technical University of Denmark (DTU) in 2017, 2018 and 2019, and one research talk at the CentraleSupélec in 2019.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cities offer plenty mobility opportunities - from an extensive public transportation system to car-sharing and bikes or scooters. People's preferences to use a city's mobility are as diverse as its range - some prefer taxi, others always take the bike, yet others combine various modes of transport (MOT). We can see that people make use of a city's multimodal nature. While the mobility offers are increasing, the time spent thinking about getting from A to B is decreasing. People rather set only cornerstones of their travel (e.g., time and location), and rely on information systems embedded in holistic mobility concepts to provide an assignment of MOTs to their travel requests. In such mobility systems the "sharing economy", which captures the societal movement away from owning towards sharing, is deeply anchored – we are facing a change from "owning cars" towards "being mobile". Moreover, sustainability considerations are indispensable when drafting future mobility. As the transport sector is one of the biggest issuer of emissions worldwide [65], decreasing the carbon footprint is crucial. Thus, urban mobility is a key topic for a sustainable future, focusing on environmentally friendly MOTs and holistic concepts.

The individually driven car is diminishing in its importance. Taking Vienna as an example, we see that the modal split of cars dropped from 31% to 25% in the last decade. Cars were the prevailing mode of transport ten years ago – today, most daily travels are covered by using public transportation in Vienna. The modal split of public transportation increased from 30% in 2010 to currently 38%, walking reaches a modal split of 30%. Moreover, even though Vienna is not considered as a prominent cycling city, already 7% of daily trips are covered by bikes [150, 151]. Besides the "traditional" modes of transport, people increasingly use sharing systems [145]. These changes and shifts to other (more environmentally friendly) MOTs come, indeed, with thought-out ideas, concepts and incentives: To start with an example, the city of Vienna reduced the price of the annual ticket for public transportation to be 1 Euro per day. This motivated more people to switch from cars to public transportation. Today, in Vienna, more people hold an annual ticket for public transportation than a car [152] and only recently the city announced that the inner city will be car-free [10]. Moreover, there is an increase in investments into bike roads, having also pop-up roads throughout the city for a further shift from cars to bikes. With this we can clearly see, that the aim is to reduce the amount of cars in the city, preferring different, environmentally friendly, MOTs. We can see ambitious and prominent mobility concepts not only in Vienna but in many cities all over the globe. The joint idea of such projects is to reduce the carbon footprint, keep the personal cars outside the cities in order to give back space to citizens, have a combined and integrated use of various modes of transport and profound sharing concepts.

By using car-sharing, resources can be employed more efficiently, it saves space and also reduces emissions. Only in Vienna, 44 million kilometer driven by car can be saved. This adds up to about 7,000 tons of CO_2 per year [102]. Studies show that only a very

small part of the privately owned cars would be needed to cover all trips taken by a car. For example, in Austria at maximum 10% of all owned cars are simultaneously driving on the roads [145] and it may vacate up to 95% of parking space [145].

Such concepts and a shift towards a "smarter" mobility is not only increasingly important in private mobility, but crucial in a corporate context too. This thesis started as a part of an applied research project, addressing these issues in a corporate context. Moreover, the latest Austrian government agreement [157] repeatedly mentions the enforcement of (shared) mobility concepts, also in a corporate context. The aim is to ecologize corporate mobility, and boost shared mobility in companies. Increasingly, companies are trying to change their view on their corporate mobility by switching from individually assigned and underutilized cars towards using sustainable and flexible mobility concepts including car pools, bike sharing, taxis, and/or public transportation services. Companies strive to have an overall green and sustainable profile and employees are aware of the importance to contribute to a greener world, even if their travel time of a trip might increase. Instead of supporting further developments in corporate mobility privileging a few selected users, we are aiming at providing sustainable corporate mobility concepts.

This thesis focuses on modeling and implementing new mobility concepts. We tackle mobility used within a closed group of users where the set of users and their mobility requests are known in advance. In our setting this closed group of users is a company with its employees but can also be, e.g., a residential area where mobility is mutually used. The aim of this thesis is to formulate relevant problem settings by means of mathematical programming, detect efficient solution techniques and develop algorithms for solving the models. Thus, we aim to tackle a multitude of questions: How can a sharing concept for a closed community be modeled? What are the important decisions, what can be taken as a given? What is an appropriate objective function to be considered? What are practical and useful extensions? How can users be incorporated in such optimization problems? What are appropriate methods to solve such problems? Can the chosen methods be of practical use? To answer these questions, we will make use of advanced modeling techniques, decomposition approaches, and mainly exact state-of-the-art algorithms. In order to obtain a realistic idea of the problems, all models are tested using generated data based on gathered statistics regarding Viennese work patterns.

1.1 Methodology

In this thesis we discuss relevant problem settings in order to support decision making for shared (corporate) mobility system. We formulate optimization problems using quantitative methods and models, and develop efficient solution techniques. In general, we distinguish between exact methods and heuristics. The latter approximates a solution whereas it is not guaranteed that the optimal solution will be found. These algorithms are usually very efficient in terms of computational effort and can find good and acceptable solutions. Exact algorithms solve the respective problem always to proven optimality, however, can be very slow.

Even though heuristics can be very powerful, we mainly focus on exact solution techniques, or a combination of exact and heuristic approaches. In order to assess the real potential of a model, one needs optimal solutions. Yet, in order to be able to solve

real-sized and practical instances, simplifications on the modeling side as well as advanced solution techniques are required.

In the following, we shortly outline the techniques used in this thesis. We start by giving an introduction into mathematical modelling, then summarize the exact approaches used in Chapter 3 and 4, that is column generation and branch-and-cut.

1.1.1 Mathematical programming

The following paragraphs in this section are based on Dantzig [48] as well as Hillier and Lieberman [83], where also further and more detailed information can be found.

Mathematical programming methods have proven to be a powerful technique. In this case, programming derives from scheduling and means a structure that is represented as a mathematical model. In such a mathematical model we aim to depict a real-world environment using mathematical terms and concepts. If all the relationships in the model are linear and if all variables are continuous, we call it a linear programming (LP) model. If some or all of the decision variables are integer, we are working on (mixed) integer programs ((M)IP). Note that in this thesis we only formulate and solve programs with linear relationships. Computational effort is then used in order to find an optimal "program", i.e., optimal solution, of given alternatives. When formulating a mathematical model it is crucial to thoroughly understand the actual problem and to know what the purpose of the study should be. There has to be a well-defined problem statement, determining the appropriate objectives, restricting constraints and interrelationships. The steps of the problem definition greatly affects the relevant conclusions.

Thus, a mathematical model is an abstract representation of a problem. The essence of the problem is formulated with parameters and constraints that limit and define the model, as well as variables which represent the decisions to be made. We maximize or minimize an objective function, e.g., cost minimization or profit maximization. Yet, models are not always restricted to one objective only, but may state multiple ones as we will see in Chapter 4. The modeling approach has a great impact on the efficiency of the problem. One has to decide what to explicitly model, which information is needed, what may be simplified. Complicated constraints can be omitted with advanced modeling techniques, as we will see in Chapter 3.

Many powerful tools exist to solve mathematical programs. A well-known commercial solver is CPLEX, that is used to solve all models in this thesis. Even though these solvers can already tackle challenging models, they reach their limits. For this, advanced solution techniques are needed. As mentioned before, in this thesis we mainly rely on two exact algorithms, shortly outlined in the following.

1.1.2 Column generation

Many works can be found on column generation. The following paragraphs are based on Lübbecke [101], where also further information can be found.

Usually, only a fraction of all variables in a LP is needed to prove optimality. Considering a problem with binary variables, at the end, many variables will take on the value 0. Column generation is a classical algorithm to solve large-sized problems by iteratively adding variables to the model. Basically, the technique exchanges information between a so called (restricted) master problem and the pricing problem.

The master problem (MP) is the to-be solved linear program including all variables and constraints. Typically, the size of variables is exponential. To start the column generation algorithm, we consider a restricted master problem (RMP), considering only a subset of variables. After solving the RMP, the optimal solution is obtained and the optimal dual solution is passed on to the pricing problem, where a variable with negative reduced cost (minimization problem) is sought. If a variable with negative reduced cost is found, we add it to the RMP. Note that at each iteration the algorithm might find, and consecutively add, multiple variables with negative reduced cost. The RMP is re-solved, the values of the optimal dual variables again handed over to the pricing problem, which is recurrently solved, aiming to detect new variables with negative reduced cost. This process is repeated until no further improving variables can be found. If so, the current solution of the RMP is the optimal solution for the MP, i.e., the linear program. Note that, considering a (mixed) integer program, this gives the lower bound of a minimization problem. If one aims to solve the program to proven integer optimality, a branch-and-price algorithm is then (usually) needed. The column generation algorithm solves the linear relaxation in each node of the branch-and-bound tree. However, in this thesis, even though we work on integer programs, we do not implement a full fledged branch-and-price algorithm but solve the integer program on the initially generated columns only, which does not necessarily give the optimal solution to the integer program.

1.1.3 Branch-and-cut

Branch-and-cut algorithms have been proven to be very powerful when solving (mixed) integer linear programs. Note that commercial solvers, such as CPLEX, use the branch-and-cut algorithm for solving models. In what follows, we shortly outline the algorithm. We refer to Mitchell [110] for more detailed annotation and information.

The algorithm is a combination of the well-known branch-and-bound algorithm and the cutting plane method. The basic idea of cutting planes is to improve the lower bound of the relaxed integer problem (assuming a minimization problem) by iteratively adding omitted constraints, which are usually either complicated constraints or of exponential size, and/or valid inequalities. Valid inequalities are constraints that do not eliminate feasible solutions but are not in the initial set of constraints. These inequalities are used to strengthen formulations of difficult integer problems.

The brach-and-cut algorithm iteratively solves the underlying model and adds violated constraints in a cutting-plane manner. To detect whether constraints are violated, separation algorithms (or routines) are called. In our algorithm in Chapter 4, we eliminate timing constraints of trips and routes, and add them as infeasible path constraints [11]. These are then separated by checking user/car routes for timing restrictions, and added as a cut if infeasible routes are detected. Afterwards, the model is resolved again and further cuts are sought. This is repeated until no further cuts can be added to the model.

Note that cuts may be added after a relaxed solution is found or only on incumbent ones. We follow the latter strategy. In latest works, depending on the problem, this has shown to be a very efficient approach [13]. The linear relaxation of our problem is enhanced by adding strong valid inequalities to the root node at the beginning of the algorithm to the model.

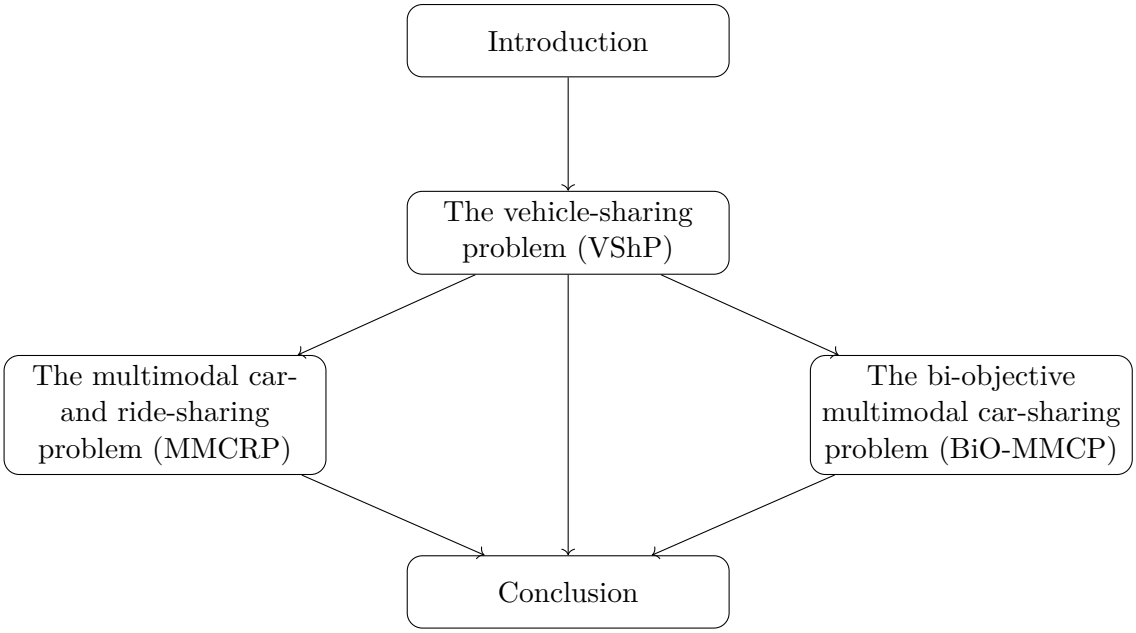


Figure 1.1: Thesis outline.

1.2 Outline and contribution

This thesis consists of three papers that have either been accepted for publication, are under review or have been submitted to a scientific journal. All papers contribute to the literature of optimization problems for shared mobility systems considering a known and closed group of users. The chapters introduce efficient modeling and algorithms to solve complex problems.

Figure 1.1 gives an overview of the thesis. We started by giving a short introduction in Chapter 1. Chapter 2 introduces "The vehicle-sharing problem (VShP)". The consecutive Chapters 3 and 4 are extensions of the first one, and tackle, additionally to car-sharing, ride-sharing as well as the incorporation of user preferences and time-dependent travel times. Lastly, we provide a conclusion in Chapter 5. All of the models and problem settings are inspired by the project "Seamless", the last model also incorporates ideas from the FWF project "MOMIP: Multi-objective (mixed) integer programming".

In the following, the individual parts of the thesis are shortly outlined and the contribution of the authors to the respective papers are stated.

Modeling and solving a vehicle-sharing problem

In Chapter 2, we study the vehicle-sharing problem in a company, having one or more offices, from which the employees (= users) have to visit various customers during office hours, e.g., for business meetings (= tasks). The time of such a task as well as the user joining the task are known in advance. This gives us fixed sequences, thus we do not determine tours but focus on the vehicle assignment. The company has a pool of shared vehicles and has access to other MOTs, such as public transportation, to cover all demanded trips. We maximize the savings obtained when using a shared vehicle instead

of any other MOT. The cost of transportation do not only include distance cost, but hourly wages of employees in order to properly reflect the trade-off between fast (but expensive) and slow (but cheap) modes of transportation. We model the problem where only one type of vehicles can be shared as the maximization equivalent of a minimum cost flow problem. The case where different vehicle types, e.g., bikes, cars and electric vehicles, are shared, the formulation is based on the multi-commodity flow problem. Even though these two problems are proven to be NP-hard, state-of-the-art solvers are able to solve the models within seconds, even for real-world sized instances. The chapter shows an efficient modeling of such sharing systems and provides a thorough analysis of savings potentials within a corporate setting.

This chapter is based on the paper "Modeling and solving a vehicle-sharing problem" submitted to the "EURO Journal of Transportation". This paper is a joint work with Prof. David Pisinger and my supervisior Prof. Sophie N. Parragh. The initial modeling idea is the output of a discussion with Prof. Pisinger. I implemented the models, conducted the literature review and the computational experiments and wrote most parts of the manuscript. This work was accompanied with advice and guidance from my co-authors regarding every part of the process.

Modeling and solving the multimodal car- and ride-sharing problem

In Chapter 3 we extend the idea of vehicle-sharing by allowing colleagues to co-ride in cars between meetings with each other, introducing the multimodal car- and ride-sharing problem. Ride-sharing is considered to have a good environmental footprint as it saves resources, such as cars and energy. In the US, on average only around 1.7 out of four seats in a car are occupied. This number even reduces to 1.2 for work-based trips [107]. The increasing number of underutilized seats in cars and an increasing number of users asking for rides, imply motivation to elaborate a sophisticated ride-sharing system. The underlying problem setting is similar to the one in Chapter 2. In addition to only sharing cars, employees may now also share rides. This will be beneficial if meetings are visited together, or different meetings are closeby or lie on the colleague's path. Moreover, the employees may now exclude some MOTs, having only a subset of MOTs available for each user. The objective is to achieve further savings by sharing cars and rides in the best possible way, thus introducing more mobility options, whilst ensuring that all customers are visited at the right time by the right employee. We enumerate all possible trips, including ride-sharing, and model them as arcs in a directed acyclic graph. Vehicle routes consist of one or more trips, whereas the driver of these trips may change at the depot. Due to the numerous possibilities, the underlying graph may be of exponential size. In order to find vehicle routes, we apply a column generation approach to solve the LP-relaxation and produce an integer solution with the generated columns, showing near-optimal solutions.

This work is published in the "European Journal of Operational Research" and is a joint work with Prof. Sophie N. Parragh, Prof. David Pisinger, and Dr. Matthias Prandtstetter. The initial motivation was provided by my co-authors with whom I specified the problem and developed the model. I implemented the master problem, the arc formulation, and the extension of the pricing strategies (different exact and heuristic approaches). I conducted all computational experiments and the literature review. The

majority of the manuscripts was also written by me. My co-authors gave advice, guidance and feedback for every part of the process.

The bi-objective multimodal car-sharing problem

In Chapter 4 we again only consider car/vehicle-sharing, but extend it by incorporating the "human factor" in a second objective. Previously, only monetary parameter were taken into account. However, when studying mobility, traveler experience is key to success and can decide on the "win or lose" of a system. Thus, convenience and user preferences are crucial. This is increasingly important as usually cost-efficiency is in conflict with a MOT's convenience. This "convenience" is difficult to measure and must be considered on an individual user level. Therefore, it is vital to tackle the trade-off between minimizing cost and enhancing the individual satisfaction of a user in a mobility system. Combining these parameters and providing the decision maker with a set of efficient solutions will lead to an enhanced acceptance of such a system. Therefore, we present the bi-objective multimodal car-sharing problem. We optimize towards two objectives, namely cost minimization and user-preference enhancement. Again, we have a set of users covering meetings. However, in this chapter we additionally study different variants where we include time-dependencies as well as not predetermined sequences of tasks and user trips. We present a branch-and-cut algorithm embedded into two bi-objective frameworks. We also also provide various heuristic solutions which, however, are not part of the submitted manuscript and are therefore presented in Appendix A.3.1.

This work has been accepted for publication at the "OR Spectrum" and is a joint work with my supervisor Prof. Sophie N. Parragh and Prof. Jakob Puchinger. I developed the models, implemented the solution algorithm, conducted the computational experiments as well as the literature review and wrote most parts of the manuscript with advice and guidance from my co-authors regarding every part of the process.

Contribution

The proposed PhD-thesis covers a variety of topics, using exact methods, efficient and new modeling approaches and incorporates single and multiple objectives. The contribution of the thesis is manifold:
In the first part, the vehicle-sharing problem is formulated as the maximization equivalent of a minimum-cost flow, as well as a multi-commodity flow problem and an extensive computational study giving managerial insights is conducted. In the second part, we introduce the novel MMCRP derived from a real-world-application, formulated as an extended vehicle sharing problem. We propose a two-layer decomposition approach, where the first layer is solved through enumeration and for the second layer we present an efficient column generation approach. With the modeling approach, we are able to hide complicated ride-sharing constraints. We show that with the proposed approach we can solve real-world sized instances to near-optimality. The computational times make it possible for a daily planning of multimodal car- and ride-sharing systems. The third and last part includes the human factor by adding a second objective function. Doing so, we can get closer to real-world requirements. We present four variants of the problem, discussing increased flexibility of the timing of the visits and time-dependent travel times and preferences. We propose a branch-and-cut algorithm, where we show that by passing

over information to subsequent iterations comes with an enhancement in computational effort.

For all chapters that correspond to already published work, I have obtained the rights to include the submitted version in my thesis. The only changes made, and the only differences to the published versions concern formatting, structural changes (renumbering of sections), and updates to cited references.

Chapter 2

Modeling and solving a vehicle-sharing problem

Submitted to: *EURO Journal on Transportation and Logistics*
July 22, 2020.
M. Enzi, S.N. Parragh & D. Pisinger.

Abstract Motivated by the change in mobility patterns, we present a scheduling approach for a vehicle-sharing problem from a company viewpoint with centralized planning. We consider vehicle-sharing in a company having one or more depots and a fixed number of users, i.e. employees. The users have appointments with a fixed location and fixed start and end times. A vehicle must be used for a full trip of a user from depot to depot. We aim at assigning vehicles to user-trips so as to maximize savings compared to other modes of transport.

We first consider that only one type of vehicles is used, and second that multiple vehicle types can be shared. For the first case we show that the vehicle-sharing problem can be formulated as a minimum-cost flow problem. Secondly, if multiple types of vehicles are available for sharing then the problem can be formulated as a multi-commodity flow problem. These formulations make the problem applicable in daily operations due to efficient solution methods.

We provide a comprehensive computational study for both cases using realistic instances based on demographic, spatial, and economic data of Vienna. We show that our formulations for this problem solve our instances in a few seconds, which makes them usable in an online booking system. In the analysis we discuss real-life cases, where we study an optimal composition of a shared fleet, restricted sets of modes of transport, and variations of the objective function.

Keywords— shared mobility, vehicle-sharing, car-sharing, transportation

2.1 Introduction

Mobility – how we use it and see it – is changing. People tend to be mobile rather than owning cars. "Mobility as a Service" (MaaS) [113] has emerged as a widely known and used term. This change is supported by novel mobility concepts, not only in the private sector but also in the area of corporate mobility. Companies are trying to change their

view on their corporate mobility by switching from individually assigned cars towards MaaS for their employees, and give incentives to use (a combination of) "greener" modes of transport to avoid pollution and congestion problems. Having shared mobility within a company (or any other closed group of users such as, e.g., home communities) will be increasingly important in future mobility settings.

In this work we study a real-life problem, and report results for relevant cases derived from an applied research project with several company partners (http://www.seamless-project.at). It is a vehicle-sharing problem in a company, having one or more offices (depots), from which the employees (users) have to visit various customers during office hours (e.g. for business meetings). Each visit (task) involves one specific user and has a fixed time, which gives us a fixed sequence of tasks. A trip covers the fixed sequence of tasks of one user, starting at a depot and terminating at the same location or in another depot of the company. Thus a trip contains several stops and it starts and ends at a predefined (but possibly different) depot. The company operates a pool of shared vehicles of a fixed size and provides possibilities to use other modes of transport (MOT), such as bikes, taxis or walk. Different to most other vehicle-sharing problems, we study the problem from a company viewpoint with centralized planning. Thus, we minimize a company's expenses and do not focus on individual goals.

The aim is to assign user-trips to the available vehicles, e.g., shared cars so as to maximize the savings obtained when using a vehicle instead of any other MOT. The costs of transportation do not only include distance cost, but also hourly wages of employees in order to properly reflect the trade-off between fast (but expensive) and possibly slower (but cheaper) modes of transport, such as public transportation or bikes. We note that cars may not always be the fastest (or cheapest) MOT. Furthermore, we also include emission cost to strengthen the use of environmental friendly MOTs.

Many formulations studying car-sharing systems are based on time-space networks such as, e.g., de Almeida Correia and Antunes [50], considering depot locations in the context of one-way car-sharing. Brandstätter et al. [36] model the movement of cars in a electric car-sharing system as a multi-commodity flow problem. Zhang et al. [154] work on vehicle-to-trips assignment and relay decisions in one-way car-sharing systems with electric vehicles. They model the problem in a single-commodity network flow model and develop a heuristic thereof. In Enzi et al. [62] both car-sharing and ride-sharing are considered simultaneously. The first step of the auxiliary graph transformation of Enzi et al. [62] is similar to the presented graph in this work. However, they extend the graph by duplicating trips including ride-sharing and solve the car- and ride-sharing problem by a kind of column generation algorithm, assigning cars to trips. Detailed surveys on car-sharing are provided by Jorge and Correia [88], Brandstätter et al. [34] and Laporte et al. [98].

In our paper, we rely on the modeling of two well-known network problems, namely the minimum-cost flow problem and the multi-commodity flow problem. Even though similar modeling approaches have been applied, we give a theoretical contribution outlining that such a shared system within a company incorporating either a single shared vehicle type or multiple shared vehicles, can be modeled using these well-known formulations. We formulate the case where only one type of vehicle is shared as a minimum-cost flow problem [4]. If more than one type of vehicles is shared, we base the formulation on the multi-commodity flow problem [18]. Note that, even though we will mainly base our

examples and results on cars, this problem can easily incorporate other shared vehicles, such as bikes or scooters.

Contribution and outline The contributions of this paper are as follows: we introduce and model a corporate vehicle-sharing problem with predetermined trips, using the well-known minimum-cost flow and multi-commodity flow formulation, which can be solved efficiently and thus used in an online operational setting within a company with centralized planning. Furthermore, we provide a detailed analysis with respect to the impact of using different kinds of shared vehicles, and provide insights into optimal fleet composition in a shared system. We also analyze the number of trips per vehicle during a day and the disadvantage (from a cost-perspective) when giving the opportunity to restrict the set of available MOTs per user/trip. We compare the case where no sharing is allowed with our introduced sharing systems. Finally, we compare the outcomes of different objective functions, whereas we once use a combination of operational distance cost and cost of time, and then considering time only.

The paper is organized as follows: We start by introducing our vehicle-sharing problem in Section 2.2. We first introduce the model with a single shared vehicle type, formulated as a minimum-cost flow problem in Section 2.2.1, followed by the model with multiple shared vehicle types formulated as a multi-commodity flow problem in Section 2.2.2. In Section 2.3 we summarize our analysis based on an extensive computational study and give managerial implications using instances based on demographic, spatial, and economic data of Vienna, Austria. We conclude this paper in Section 2.4.

2.2 A vehicle-sharing problem

Formally, our vehicle-sharing problem can be formulated as follows:

We have a set of users P that have to visit meetings (tasks). Each task is associated with a different location and has an associated fixed start time and duration. The user-to-task assignment is not interchangeable, resulting in a fixed sequence of tasks per user. Every user $p \in P$ covers one or more trips π. A trip has an origin o_π and destination e_π whilst covering in between a fixed set of tasks. Moreover, we consider a set of modes of transport K such as walk, bikes, public transportation (bus, train, metro), taxis and cars, where at least one MOT k has a restricted capacity that is shared, e.g., cars. If a trip is started with one mode of transport, then it should be used for the full trip.

Let us assume a task q and its fixed successor q'. We know the driving time between the two tasks (q, q') using mode of transport k, and the cost of driving between two tasks (q, q') using mode of transport k. Since the trips follow a fixed sequence, we can calculate cost and travel time for each trip π and each MOT k.

For each trip π let $\min_{k \in K \setminus \{1\}} C_\pi^k$ be the cost of the cheapest mobility types excluding cars $k = 1$ (assuming that we are sharing cars). Let C_π^1 be the cost of riding the same trip π by car $k = 1$. We then calculate the savings $s_\pi = C_\pi^{K \setminus \{1\}} - C_\pi^1$ of using a car compared to using the cheapest possible other mobility type. Note that if traveling with a certain MOT is not possible, we impose a penalty and set $C^k = \infty$.

Finally, we aim at assigning user-trips to the shared vehicles in the best possible way whilst maximizing savings obtained when using a car compared to the cheapest other mobility type.

We model the problems on a directed acyclic graph (DAG). Since a MOT must be used for the full trip, we do not model the tasks covered by a trip in the graph, and only consider nodes o_π and e_π for each trip π, which represent starting and ending points of a trip. The savings of the arc (o_π, e_π) is s_π, as explained above. In order to connect the trips we insert additional arcs $(e_\pi, o_{\pi'})$ if trip π_π has the same destination as trip $\pi_{\pi'}$ has origin, and the trip π finishes before the trip π'. The savings of such an arc is 0.

In the following, we introduce the modeling of the two cases presented in this paper. First, we introduce the modeling approach for the case where only one type of vehicles is shared and then solved as a minimum-cost flow problem. Second, we present the formulation where multiple shared vehicle types can be employed. This is then modeled and solved as a multi-commodity flow problem.

2.2.1 The vehicle-sharing problem with a single type of shared vehicle (*VShP-1T*)

For the vehicle-sharing problem with a single type of shared vehicle (*VShP-1T*) we create a node A_d for each depot $d \in D$ with a supply δ_d representing the number of available vehicles. Depots represent locations where the shared vehicles start and end, e.g. a company's offices. For each depot d where the vehicles must be parked in the evening, we create a node A'_d with a demand δ'_d equal to the number of requested vehicles at the end of the planning horizon. Every node A_d is connected to all nodes o_π if trip π starts in depot d. Every node e_π is connected to node A'_d if the trip π ends in depot d. We add extra arcs (A_d, A'_d) with infinite capacity and zero savings, to represent the case where a vehicle is not used and stays in the depot. Finally, we draw the nodes in a time-space network, where the x-axis represents the time of day, and the y-axis represents the depots.

Figure 2.1 shows a simple example in which we have two depots, and five trips. We assume that the first depot has two vehicles available in the morning, and two vehicles (not necessarily the same) should be returned to the depot in the evening. Note that we indicate the savings and capacity for each arc in the form (*savings, capacity*).

Let V be the set of all nodes and let s_{ij} be the savings of a trip going from node i to j (in our auxiliary graph e_π, o_π). Furthermore, let δ_i be the demand at the depots, being 0 for e_π and o_π. Parameter u_{ij} gives the capacity of an arc, which is 1 for all arcs. Finally, the binary decision variables x_{ij} take on value 1 if connection (i, j) is chosen, and 0 otherwise.

With this, we show that the vehicle-sharing problem considering one single type of shared vehicle (*VShP-1T*) can be modeled as the maximization equivalent to a minimum-cost flow problem, formulated in model (2.1)-(2.4).

$$\max \quad \sum_{(i,j)\in V} s_{ij} x_{ij} \tag{2.1}$$

$$\text{s.t} \quad \sum_{j\in V} x_{ji} - \sum_{j\in V} x_{ij} = \delta_i \quad \forall i \in V \tag{2.2}$$

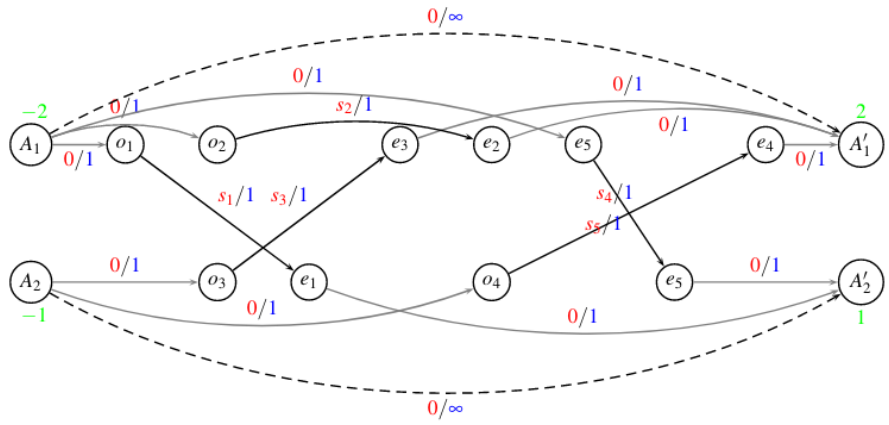


Figure 2.1: The underlying graph of the minimum-cost flow formulation of the vehicle-sharing problem with one shared vehicle type, five trips, and two depots. Nodes A_d, A'_d represent depots where the available vehicles are stored at the beginning and end of the time horizon. In our example we have $\delta_1 = 2$ vehicles available at A_1 and $\delta_2 = 1$ car at A_2, the same amount of vehicles has to be returned in the evening to A'_1 and A'_2. Nodes o_π and e_π give start and end points of a trip π. Finally, each arc represents a trip π with a given saving s_π and capacity. The x-axis represents the time of day, and the y-axis represents the depots.

$$x_{ij} \leq u_{ij} \qquad \forall i,j \in V \tag{2.3}$$
$$x_{ij} \geq 0 \qquad \forall i,j \in V \tag{2.4}$$

The objective function (2.1) maximizes savings. Constraint (2.2) restricts the out/ingoing vehicles at the beginning/end of the day. Further it assures flow conservation in nodes $i \in V \setminus \{D\}$. Constraint (2.3) makes sure that at most one vehicle is covering a certain connection (i,j).

We will solve our model as a mixed integer program (MIP) since state-of-the-art solvers are already capable of handling these kinds of problems very efficiently. Nevertheless, we shortly review some of the algorithms that have been widely applied. Ford and Fulkerson [68] were first to introduce a combinatorial algorithm for the problem. Edmonds and Karp [57] proposed the scaling resulting in the first weakly polynomial-time algorithm. Tardos [138] introduced the minimum cost circulation algorithm which was the first strongly polynomial method. In the consecutive years many solution approaches evolved. Scaling techniques have shown to be promising [57, 75, 74, 40]. Polynomial in time are also cycle cancelling algorithms [93, 73] or cut cancelling algorithms [63]. Furthermore, the network simplex method was efficiently applied to the maximum flow problem [49, 92, 100] or adaptions of the successive shortest path algorithm [37]. Kovács [96] provides a survey of various algorithms and present an overview of their respective complexity.

2.2.2 The vehicle-sharing problem with multiple types of shared vehicles (*VShP-xT*)

In what follows, we do not only consider one type of shared vehicle but multiple ones. Note that shared vehicles can be different types of cars but also bikes or any other MOT.

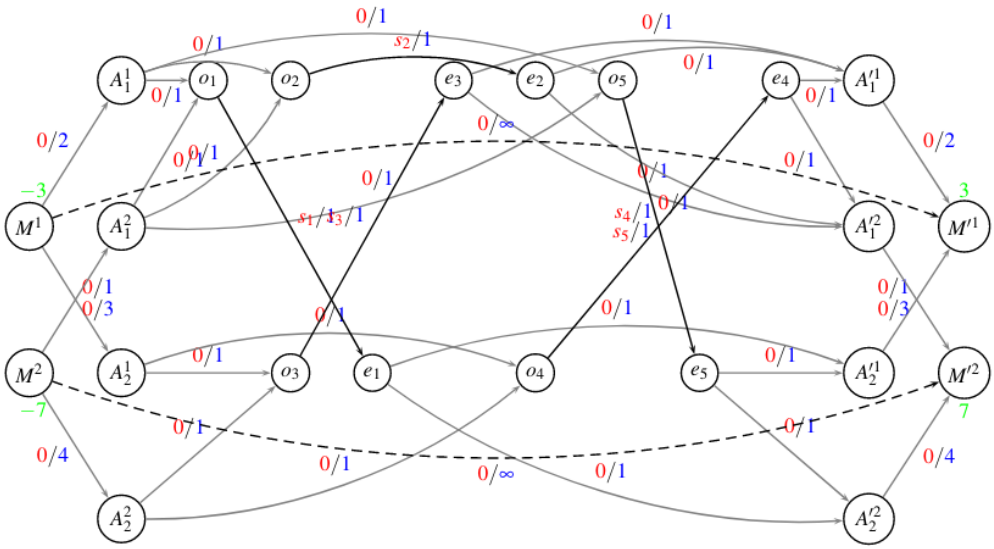


Figure 2.2: The underlying graph of the multi-commodity flow formulation of the vehicle-sharing problem with two shared types of vehicles, five trips π, and two depots d. Nodes M^k, M'^k represent supra-nodes where the available shared vehicles are stored at the beginning and end of the time horizon and then distributed to the respective depot nodes $A_d^k, A_d'^k$. We have 3 vehicles of type 1 available and 7 vehicles of type 2; 2 and 1 of type 1 are distributed to depot 1 and 2, respectively; 3 and 4 of type 2 vehicles to depot 1 and 2, respectively. Nodes o_π and e_π give start and end points of a trip π. Finally, each arc gives its respective savings and capacity. The x-axis represents the time of day, and the y-axis represents the depots.

We start with the previously described graph. To model the vehicle-sharing problem with multiple types of shared vehicles (*VShP-xT*), we duplicate the sources and sinks since we have different MOT options and supra-nodes where the MOTs start/end. We model a supra-source M^k for each $k \in K'$ where $K' \in K$ denote the set of shared vehicles. In our example we have M^1 for one type of cars, and a supra-source M^2 for another type. In a similar way we add supra-sinks M'^k. The set of all supra-nodes, thus $M^k \cup M'^k$, is denoted as M. We then construct start and end depot nodes $A_d^k, A_d'^k$ to where we connect the respective M^k and all origins o_π and end nodes e_π of a trip π, respectively. We assign savings s and capacity to each trip, i.e. arc. Drawing the nodes in a time-space network, Figure 2.2 shows a simple case where we have two shared types of vehicle.

We show that the problem can then be solved as an integer multi-commodity flow problem, where arc savings s_{ij}^k depend on the commodity transported k. In our example commodities correspond to different shared vehicles. We now consider a demand Δ_k per MOT k and define with the capacity parameter u_{ij} how many MOTs are available between two nodes i and j. Let x_{ij}^k be 1 if connection between (i,j) is covered by MOT k, 0 otherwise. This problem has the formulation:

$$\max \quad \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} s_{ij}^k x_{ij}^k \tag{2.5}$$

$$\text{s.t} \quad \sum_{i \in V} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall k \in K, j \in V \setminus \{M\} \tag{2.6}$$

$$\sum_{j \in V} x_{i,j}^k - \sum_{j \in V} x_{j,i}^k = \Delta_k \quad \forall i \in M^k, k \in K \tag{2.7}$$

$$\sum_{i \in V} x_{i,j}^k - \sum_{i \in V} x_{j,i}^k = \Delta_k \quad \forall j \in M'^k, k \in K \tag{2.8}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} \quad \forall i, j \in V \tag{2.9}$$

$$x_{ij}^k \geq 0 \quad \forall k \in K, i, j \in V \tag{2.10}$$

Objective function (2.5) maximizes the savings. Equation (2.6) gives the flow conservation constraints for all nodes except the sources and sinks. Constraints (2.7) and (2.8) restrict the number of shared MOTs. Constraint (2.9) gives the capacity restriction on the arcs. Finally, constraint (2.10) assures positive numbers.

The formulation above is polynomial in the size of the constraints, having $|K| \cdot |E|$ variables, where $|E|$ is the number of arcs, and $|E| + |K| \cdot |V|$ constraints. However, large-scale problems may be challenging to be solved. Therefore, efficient solution algorithm have been applied such as Lagrangian relaxation [124, 12], adapted branch-and-bound algorithms [17], Dantzig-Wolfe decomposition [90] and column generation algorithms [139, 16]. Nevertheless, we will solve the models as a MIP as state-of-the art commercial solvers are able to solve problems of limited size (like in our case) within seconds.

2.3 Computational results

We provide computational results using the above presented models for the vehicle-sharing problem. The models are implemented in C++ and solved with CPLEX 12.9. Tests are carried out using one core of an Intel Xeon Processor E5-2670 v2 machine with 2.50 GHz running Linux CentOS 6.5. Tests are conducted on a number of generated instances varying in size and complexity.

In the following, we give a short introduction to the instance set. Afterwards we provide the results of our computational study for the *VShP-1T* and *VShP-xT*. We further present results of varying objective functions and restricted sets of MOTs for individual users. Lastly, we comment on the results and give some managerial insights.

2.3.1 Test instances

We generate realistic benchmark instances based on available demographic, spatial and economic data of the city of Vienna, Austria. Five different MOTs are considered: cars (combustion engine vehicles and electric vehicles), walk, bike, public transportation and taxi. In the following results we name the combustion engine car 'car-type1', the electric vehicles 'car-type2'. For each mode of transport $k \in K$ we define distances, time and cost between all nodes. We calculate the Aerial distance between two locations which are then multiplied by a constant sloping factor for each MOT k in order to account for longer/shorter distances of the respective mode of transport. Moreover, we have emissions per distance unit, average speed, cost per distance and cost per time as well as additional time needed for, e.g., parking the car, for each $k \in K$. The cost of time is a fixed value based on the average gross salary including additional costs for employers in

Austria. The objective function results from these values. The values of the parameters are given in the Appendix A.1 Table A1.

Each generated instance represents a distinct company operating two offices and consisting of a predefined set of users, i.e. employees, $p \in P$. The locations of the offices (depots) are based on statistical data of office locations in Vienna placed in the geometric centers of all 250 registration districts.

Companies are defined by a fixed number of users u. Note that one person may have more than one trip assigned. Therefore, the number of users u does not equal the number of trips (arcs) in the graph. In Table 2.1 we provide an overview on the average number of trips per user. As we can see, on average each user takes about 1.5 trips during the planning horizon.

The number of meetings and their time and location, are randomly generated based on historic statistical data. We define a time horizon of one day where each user has an assigned set of meetings distributed over the day. We calculate savings based on the cheapest other MOT, whereas we always use publicly available MOTs (public transportation, bike, taxi) to be the cheapest other possible alternative.

We solve 10 instances per instance group.

Table 2.1: Average number of trips for each instance group with u users.

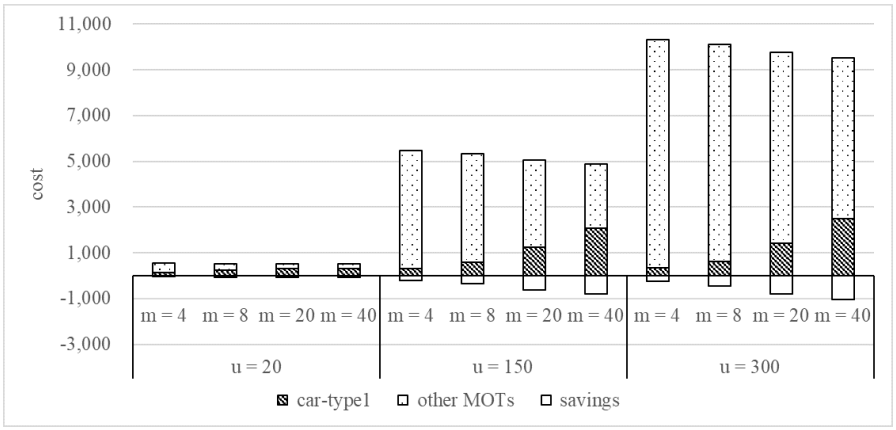
u	*20*	*50*	*100*	*150*	*200*	*250*	*300*
trips	31	76	147	218	287	358	427
trips / u	1.54	1.52	1.47	1.45	1.44	1.43	1.42

A more detailed instance description can be found in Enzi et al. [62]. Knopp et al. [95] base their instance generation on the same idea, and provide a detailed description at the end of their paper. Instance sets are made publicly available at `https://github.com/dts-ait/seamless`.

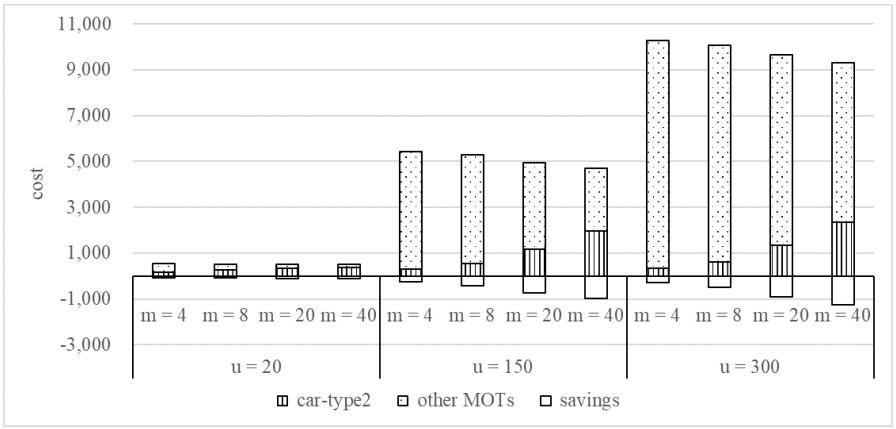
2.3.2 Results for the vehicle-sharing problem with a single type of shared vehicle (*VShP-1T*)

We start by showing the results obtained for the *VShP-1T*, represented by model (1)-(4). We assume one type of shared vehicle: in *VShP-1T:car* these are combustion engine cars (car-type1), in *VShP-1T:ecar* we consider electric cars (car-type2) as our shared resource. The results are obtained for an increasing number of users u, and a varying number of shared cars m. Walk, bike, public transportation, and taxi are assumed to have no capacity restriction. The considered cars are equally spread over the two depots.

Figure 2.3 illustrates the average total cost as a sum of the cost of the shared cars and other MOTs as well as savings for users $u = 20, 150, 300$ and number of cars cars $m = 4, 8, 20, 40$. Note that the savings are given in the opposite direction (negative numbers) for a better distinction between savings and cost. With an increase in the number of shared cars m we see a decrease in the overall costs, clearly visible by the declining bars in each group. For smaller instances ($u = 20$) we can observe that the cost of the cars is higher than the cost of the other MOTs. This is not surprising as the model is able to assign the shared cars to all beneficial trips. Figure 2.3(a) shows the values for



(a) *VShP-1T:car*



(b) *VShP-1T:ecar*

Figure 2.3: Total cost split into cost of MOTs and cost of cars, respectively for car-type1 (=combustion engine cars) and car-type2 (=electric cars) as well as savings (negative bars) for an increasing number of $u = 20, 150, 300$ and cars $m = 4, 8, 20, 40$.

VShP-1T:car, Figure 2.3(b) the *VShP-1T:ecar*. As can be seen, the general impression as well as the total overall cost are about the same. In Figure 2.3(b), thus for the case where electric vehicles are shared, we have slightly less total cost and less car cost. More detailed information and further results on, e.g., savings, and the composition of the total cost regarding cars and other MOTs, can be found in Appendix A.1 Table A2 and Table A3.

Tables 2.2 and 2.3 summarize the average number of trips for *VShP-1T:car* and *VShP-1T:ecar* and increasing number of cars m and users u. We observe that with an increasing number of users u the average number of trips for a car is also raising. This is because the model aims to cover as many trips by car as possible. With an increasing number of users but the same number of cars in the system, the model will try to situate more trips on one of the few cars available. The average number of trips is higher when fewer cars are available. We observe this for both variants, the *VShP-1T:car* and *VShP-1T:ecar*. Overall *VShP-1T:ecar* shows a higher average of trips per car.

Table 2.2: Average number of trips per car for an increasing number of users u and cars m for *VShP-1T:car*.

m	u=	20	50	100	150	200	250	300
4		1.5	2.1	2.1	2.2	2.3	2.5	2.3
8		1.4	1.9	2.1	2.1	2.2	2.3	2.1
20		1.3	1.6	1.9	1.9	1.9	2.1	2.0
40		1.4	1.4	1.6	1.7	1.8	1.9	1.9

Table 2.3: Average number of trips per car for an increasing number of users u and cars m for *VShP-1T:ecar*.

m	u=	20	50	100	150	200	250	300
4		1.6	2.1	2.2	2.3	2.4	2.7	2.4
8		1.5	2.0	2.2	2.2	2.2	2.4	2.2
20		1.3	1.7	2.0	1.9	2.0	2.2	2.1
40		1.4	1.4	1.8	1.8	1.9	2.0	2.0

In Appendix A.1 Table A5 we give an overview of the solution times for *VShP-1T:car* and *VShP-1T:ecar*. For an increasing number of users u, we observe an increase in the times used to solve the models. However, we always stay below 8 seconds of solution time.

2.3.3 Results for the vehicle-sharing problem with multiple types of shared vehicles (*VShP-xT*)

In the following, we present the results obtained by solving the *VShP-xT*, given in model (5)-(10). We now assume different types of shared vehicles in one model. For our tests we use combustion engine cars (car-type1) and electric vehicles (car-type2). Note that this can be easily extended/changed in order to include, e.g., bikes or e-scooters. We are given an equal number of each car type, denoted as m_k respectively. Thus if $m_k = 2$, then two cars of each type are available. These are then again equally assigned to the depots. In our example, since we assume 2 depots, this would give us one combustion engine car (car-type1) and one electric vehicle (car-type2) at each depot. Again, the model is tested on a number of instances for a varying number of users u and shared cars m.

Figure 2.4 plots the total cost as a result of the cost of the two car types, and cost of the other MOTs, as well as savings which are given in the opposite direction as negative numbers. Note that we observe a similar picture as in Figure 2.3. We increase the cost of cars (car-type1 and car-type2) by adding more available cars m to the system whilst reducing total cost. The share of the car-type2 cost are constantly higher than the cost of car-type1. This means, as e-cars (= car-type2) are usually cheaper, that more electric cars are assigned. Note that for the smallest instances ($u = 20$) the cost of the car-type1 is diminishing, meaning that almost all of the trips are covered by car-type2. Table A4 in the Appendix A.1 shows more details on the cost as well as the breakdown of the total cost into cost of the respective car types and other MOTs.

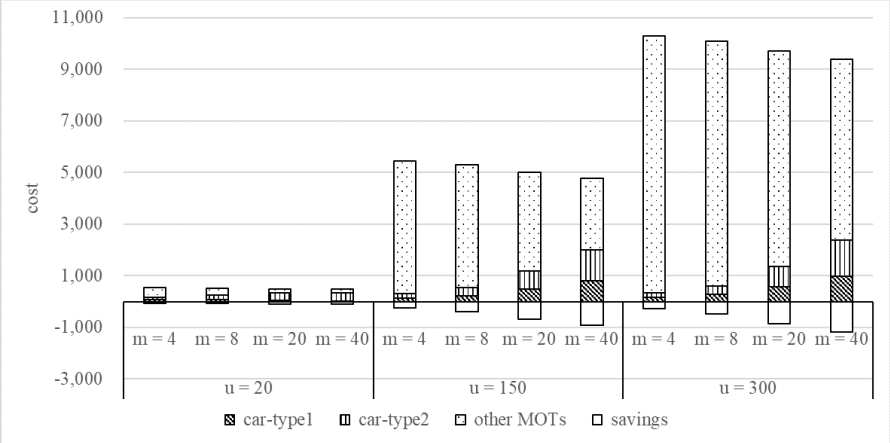


Figure 2.4: Total cost split into cost of MOTs, cost of car-type1 (=combustion engine cars), and cost of car-type2 (=electric cars) as well as savings (negative bars) comparison for $u = 20, 150, 300$ and cars $m = 4, 8, 20, 40$ for *VShP-xT*.

Table 2.4 shows the average number of trips a car is taking for an increasing number of users u and cars m. The results are split into values for the different car types. We can see that the average number of trips per car-type2 (= electric car) is always greater than the number of trips for the other type. This means, that if possible, the model aims to assign more trips to e-cars. In the extremest case ($u = 20$,$m = 40$) almost no trip is covered by a conventional car (car-type1). Moreover, we can again observe an increase in the average number of trips per car for a higher number of users u as well as smaller number of cars m.

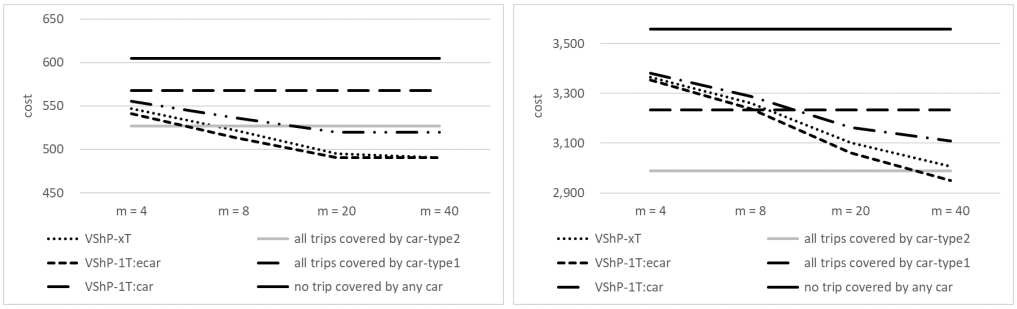
Table 2.4: Average number of trips per car for an increasing number of users u and cars m for *VShP-xT*. Car-type1 are combustion engine cars, car-type2 electric cars.

m		u=	20	50	100	150	200	250	*300*
4	car-type1		1.5	1.7	1.7	1.9	2.1	2.1	1.9
	car-type2		1.8	2.5	2.6	2.6	2.6	3.0	2.8
8	car-type1		1.3	1.6	1.7	1.6	1.8	1.8	1.7
	car-type2		1.6	2.3	2.6	2.6	2.7	2.8	2.6
20	car-type1		0.4	1.2	1.6	1.4	1.4	1.6	1.5
	car-type2		1.5	2.1	2.4	2.4	2.5	2.7	2.5
40	car-type1		0.1	0.7	1.3	1.3	1.4	1.4	1.4
	car-type2		1.3	1.7	2.1	2.1	2.3	2.5	2.4

All instances can be solved in less than 17 seconds of solution time (see Appendix A.1 Table A5).

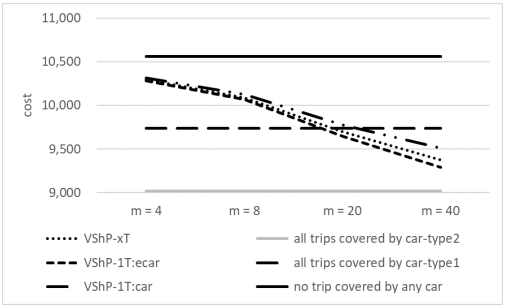
2.3.4 Comparison of *VShP-1T:car*, *VShP-1T:ecar*, *VShP-xT* and the case where all trips are covered by one vehicle/MOT

Now we compare the models *VShP-1T:car*, *VShP-1T:ecar*, and *VShP-xT* regarding cost for an increasing number of users u and vehicles m.



(a) u = 20

(b) u = 100



(c) u = 300

Figure 2.5: Total cost comparison where all trips are either covered by electric cars (car-type2), combustion engine cars (car-type1) or not by cars at all, and the introduced models (*VShP-1T:car*, *VShP-1T:ecar*, *VShP-xT*). The restricted fleet (given on the x-axis, $m = 4, 8, 20, 40$) is only applicable to the cases where vehicles are shared (*VShP-1T:car*, *VShP-1T:ecar*, *VShP-xT*) as otherwise all trips are covered by the respective MOT.

Figure 2.5 shows the cost of the different cases for $u = 20, 100, 300$ and increasing m. The respective lines give the cost of the following cases: no trip is covered by a car, every trip is covered by a combustion engine vehicle (car-type1), all trips are covered by electric vehicles (car-type2), *VShP-1T:car*, *VShP-1T:ecar* and *VShP-xT*. Note that the fleet restrictions only apply for *VShP-1T:car*, *VShP-1T:ecar* and *VShP-xT*. We can see that it is always most expensive if no trip is covered by a car. In all three figures, the line representing cost of using only car-type2, which are electric vehicles, lies below the line showing cost when using car-type1, thus combustion engine vehicles, only. When considering $u = 20$, *VShP-1T:car*, *VShP-1T:ecar*, and *VShP-xT* are always cheaper than employing conventional cars only. For $u = 100$ and $u = 300$ the cost curves of the three models (*VShP-1T:car*, *VShP-1T:ecar*, *VShP-xT*) start above the conventional car cost, however break even after $m = 8$ for $u = 100$ and around $m = 20$ for $u = 300$. In Figure 2.5(a) the cost of the three models cross the line representing the cost if all trips are covered by car-type2. For $u = 100$ (Figure 2.5(b)) the cost line representing *VShP-1T:ecar* crosses the line where only electric cars are employed at around $m = 40$. For $u = 20$ the *VShP-xT* merges at some point with *VShP-1T:ecar*, as there are enough electric vehicles to cover all beneficial trips. Overall, *VShP-1T:ecar* is always the cheapest option. The cost line of *VShP-1T:car* is always above *VShP-xT* and *VShP-1T:ecar*. Thus, considering the three sharing options, employing a shared system with only combustion engine cars (*VShP-1T:car*) is most expensive.

Table 2.5 compares the cost of the three models *VShP-1T:car*, *VShP-xT*, and *VShP-1T:ecar*, whereas the latter is taken as the base. We show the averages over $m = 4, 8, 20, 40$ for an increasing number of users u. *VShP-1T:ecar* is the cheapest, as we have already seen in the previously discussed figure. *VShP-xT* shows on average slightly higher cost. Lastly, as expected, the case where only combustion engine cars are employed in a pool of shared vehicles, is the most expensive alternative, ranging up to 1.05 times the cost of *VShP-1T:ecar*.

Table 2.5: Total cost comparison split for an increasing u and averaged over all m for *VShP-1T:car*, *VShP-1T:ecar*, and *VShP-xT*. Column 'cost' gives the absolute cost of the respective model, 'comp.' compares the cost to *VShP-1T:ecar* where it is set as (cost of the model / cost of *VShP-1T:car*).

	VShP-1T:ecar	*VShP-xT*		*VShP-1T:car*	
u	cost	cost	comp.	cost	comp.
20	509	514	1.01	533	1.05
50	1,541	1,558	1.01	1,599	1.04
100	3,152	3,184	1.01	3,235	1.03
150	5,091	5,127	1.01	5,184	1.02
200	6,962	7,003	1.01	7,065	1.01
250	8,909	8,954	1.00	9,022	1.01
300	9,822	9,866	1.00	9,932	1.01

2.3.5 Including user preferences as a restricted subset

We assume that every user p has a set $K^p \subseteq K$ of possible modes of transport that can be used, reflecting her preferences. Depending on the user that is covering a trip π, we can then define a set of modes of transport possible to be assigned for a trip $K^\pi \subseteq K$. Note that if a MOT is not in the respective set K^π we impose a penalty and set $C^k = \infty$. We define seven different cases aiming to represent differences in preference distribution.

For the first case, prefVar0, we make use of available statistical data representing the working population of Vienna. For this we define different combinations of possible accepted MOTs in the instance generation: generic, motorised only, no public transportation, no motorised, cars only, public transportation only and bike only. For each of them we have a probability for female and male users, where we have [0.19, 0.03, 0.01, 0.04, 0.18, 0.42, 0.13] and [0.18, 0.03, 0.02, 0.03, 0.26, 0.35, 0.13], respectively [132]. We assume that 53% of the working population is male, and 47% female [134]. Further, we incorporate the probability that 87% of them have a driving license and 13% are not allowed to drive a car [27]. The combinations are then chosen randomly based on the set probability distribution. We assume that if a user includes a combustion engine car in her set of MOTs, then she will also have the electric car and vice versa. For the other cases, naming prefVar1-prefVar6, we adopt more straightforward strategies to represent the preferences of the users. Depending on the variant, we define a fixed percentage of users with a given setting. We say this may either be mixed (= accepting all MOTs), cars only or other MOTs except cars (= no cars). Let us assume an instance with 20 users and 40% mixed, 40% cars only and 20% other MOTs only. Then users 1-8 accept

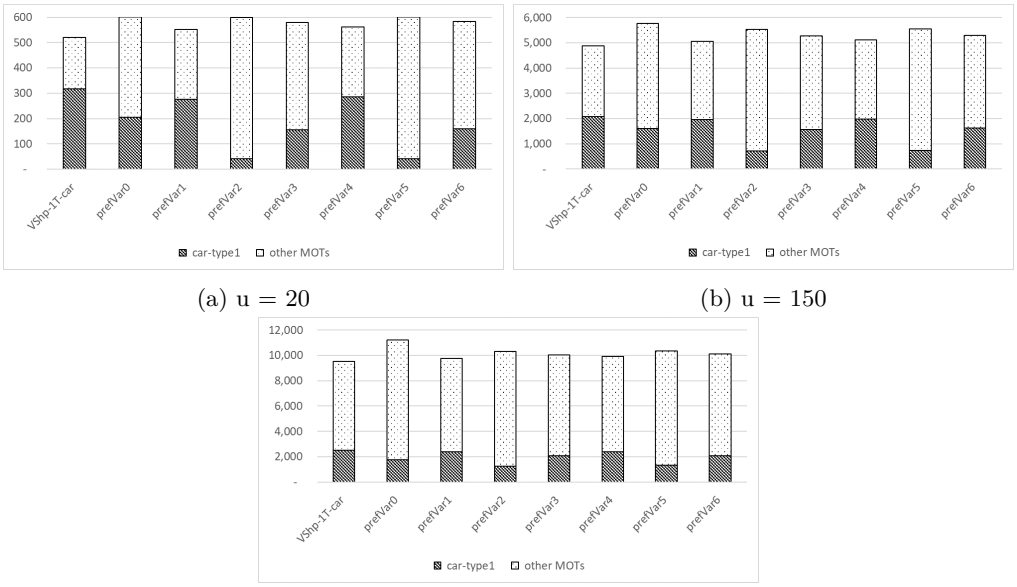
Table 2.6: Categorization of the different preference variants. Percentage of the users with the respective set of accepted MOTs, where (1) all: no restricted set is applied, user takes all MOTs, (2) cars only: the user only wants to drive by car, (3) no cars: no cars are given in the restricted set, only other MOTs are accepted.

variant	all	cars only	no cars
prefVar0		see text	
prefVar1	40%	40%	20%
prefVar2	10%	10%	80%
prefVar3	25%	25%	50%
prefVar4	0%	80%	20%
prefVar5	0%	20%	80%
prefVar6	0%	50%	50%

all MOTs, users 9-16 only cars and users 17-20 anything but cars. Table 2.6 shows the setting of each of the applied variants.

Figure 2.6 shows the total cost divided into cost of cars and other MOTs for *VShP-1T:car* for an increasing number of users $u = 20, 150, 300$ and cars $m = 40$ for the different preference settings, namely prefVar0-prefVar6. The first bar in every subfigure gives the respective cost of the base case where no restricted set of preferred MOTs is given, which is *VShP-1T:car*. For $u = 20$ in Figure 2.6(a) we can still see a substantial difference in the composition of the cost of the different settings, yet similar total cost. It is clearly visible, that *VShP-1T:car* without any restricted set of MOTs, outputs the least cost, however, have the highest cost for operating cars. The cost for cars take up about 60% of the total cost. The lowest share of cost for cars is used in prefVar3, only having 27% of car cost compared to total cost. The difference between the cheapest and most expensive variant is about 13%. We cannot observe a big difference for giving mixed preference setting and restricting to cars only (comparing prefVar1 and prefVar4, prefVar2 and prefVar5, prefVar4 and prefVar6). Similar pictures are given for $u = 150$ and $u = 300$ in Figures 2.6(b)(c). For $u = 150$ we see a difference in total cost between the cheapest and most expensive one, which is about 15%, and for $u = 300$ the difference between the extremes in terms of total cost is 15%. The most expensive variant in both instances is prefVar0. The lowest/highest share of car cost is 13/43% for $u = 150$ and 12/26% for $u = 300$.

In Table 2.7 we compare the cost of each variant with the *VShP-1T:car*. For each variant (prefVar0-prefVar6) we show the average cost of using conventional cars (car-type1), cost used for all other MOTs and in total for $u = 300$ and $m = 40$. We also have a second column for each variant, stated as "comp.", where we compare the cost to the base case calculated as (cost of the variant / cost of *VShP-1T:car*). We see that our base case is the most expensive regarding car usage. In prefVar2, where most of the users prefer all MOTs except cars, we only use 0.59 times the cost of cars compared to the *VShP-1T:car*. Conversely, regarding other MOTs, the simple *VShP-1T:car* is the cheapest variant, where prevVar0 uses 1.39 times more the cost on average. This comparably big difference in cost is mainly attributable to the more subtle differentiation of the preference settings. As in prefVar0 we also distinguish whether a person would, e.g., only take public transportation. In total we confirm the picture from above, that *VShP-1T:car*



(a) u = 20

(b) u = 150

(c) u = 300

Figure 2.6: Total cost split into cost of MOTs and cost of cars (car-type1) for $u = 20, 150, 300$ and $m = 40$ for different variants of MOT-preference settings and *VShP-1T:car*.

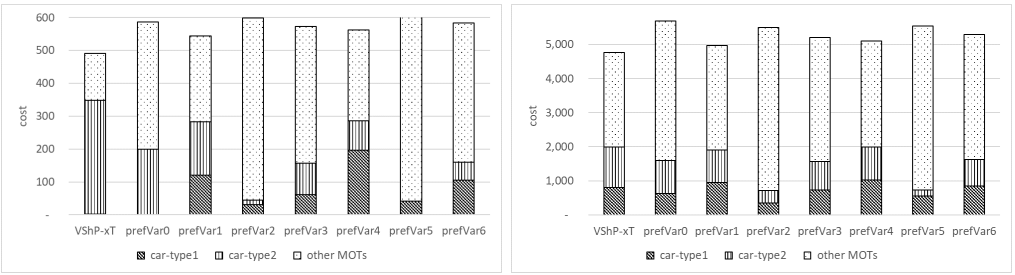
without any restriction, is the cheapest setting, however prefVar1 or prefVar3 only have 1.03 times the cost. Further results can be found in Table A6 in the Appendix A.1.

Table 2.7: Total cost comparison split into cost of car-type1 (=combustion engine cars) and other MOTs for $u = 300$ and $m = 40$ for different variants of MOT-preference settings and *VShP-1T:car*. Column 'cost' gives the absolute cost of the respective variant, 'comp.' compares the cost to *VShP-1T:car* where it is set as (cost of the variant / cost of *VShP-1T:car*).

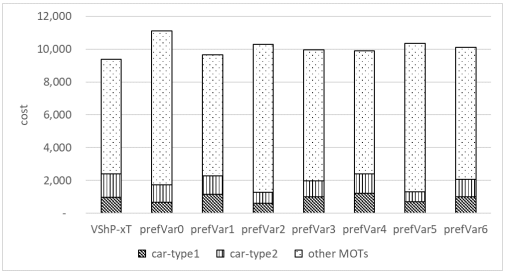
	VShP-1T:car	prefVar0		prefVar1		prefVar2		prefVar3		prefVar4		prefVar5		prefVar6	
	cost	cost	comp.	cost	comp.	cost	comp.	cost	comp.	cost	comp.	cost	comp.	cost	comp.
car-type1	1,227	872	0.71	1,169	0.95	722	0.59	1,030	0.84	1,188	0.97	747	0.61	1,045	0.85
other MOTs	8,704	12,093	1.39	9,014	1.04	9,669	1.11	9,231	1.06	9,220	1.06	9,682	1.11	9,337	1.07
total	9,932	12,964	1.31	10,184	1.03	10,391	1.05	10,261	1.03	10,408	1.05	10,429	1.05	10,382	1.05

Figure 2.7 shows the differences in cost of the different preference settings when solving *VShP-xT*. Figure 2.7(a) gives the averages for $u = 20$, Figure 2.7(b) for $u = 150$ and Figure 2.7(c) represents $u = 300$. The number of shared cars is always set to $m = 40$. We again observe structural differences between the variants, however now also between the similar variants (prefVar1 and prefVar4, prefVar2 and prefVar5, prefVar4 and prefVar6). In Figure 2.7(a) the base case and prefVar0 are covered by electric cars (car-type2) and other MOTs only. The other variants employ a mix of conventional cars and electric cars. This is then also visible for *VShP-xT* and prefVar0 in Figure 2.7(b)(c).

Table 2.8 summarizes the average cost of the basic setting (*VShP-xT*), and prefVar0-prefVar6 partitioned into cost for combustion engine cars (car-type1), electric cars (car-type2), other MOTs and in total for $u = 300$ and $m = 40$. For each variant we again show the cost and the comparison (comp.) to the *VShP-xT* calculated as (cost of the



(a) u = 20 (b) u = 150



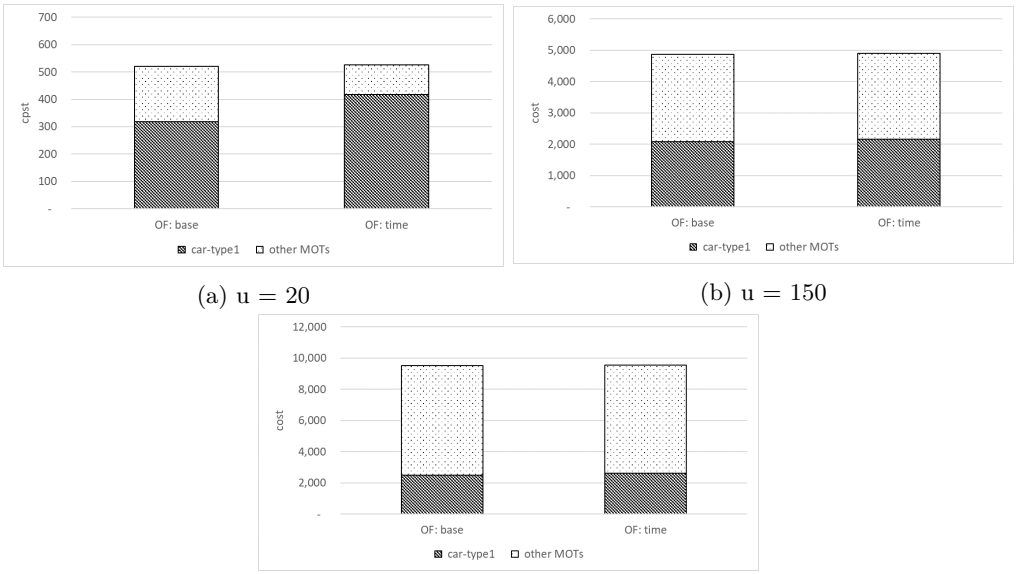
(c) u = 300

Figure 2.7: Total cost split into cost of MOTs and cost of car-type1 and car-type2 (both combustion engine and electric cars) for $u = 20, 150, 300$ and $m = 40$ for different variants of MOT-preference setting and *VShP-xT*.

variant / cost of *VShP-xT*). Here we have a somewhat different picture than above. *VShP-xT* is still the most cost efficient in total, with prefVar0 using up to 1.31 times the cost. Again, this comparably big difference in cost is mainly attributable to the more subtle differentiation of the preference settings. For car-type2, which is electric cars, the base case where no preferences are taken, is the most expensive one, as can be seen that all numbers of the 'comp.' columns are below 1. Comparing the cost of car-type1, this differentiates. For some cases (prefVar1, prefVar3, prefVar4, prefVar6) the cost are higher or equal to *VShP-xT*. For the others the results show lower cost, e.g., prefVar0 only 0.69 of total cost. Further results can be found in Table A7 in the Appendix A.1.

Table 2.8: Total cost comparison split into cost of cost car-type1 and car-type2 (=combustion engine car and electric car), and other MOTs for $u = 300$ and $m = 40$ for different variants of MOT-preference settings and *VShP-xT*. Column 'cost' gives the absolute cost of the respective variant, 'comp.' compares the cost to *VShP-xT* where it is set as (cost of the variant / cost of *VShP-xT*).

	VShP-xT	prefVar0		prefVar1		prefVar2		prefVar3		prefVar4		prefVar5		prefVar6	
	cost	cost	comp.	cost	comp.	cost	comp.	cost	comp.	cost	comp.	cost	comp.	cost	comp.
car-type1	491	339	0.69	558	1.14	348	0.71	491	1.00	593	1.21	383	0.78	515	1.05
car-type2	683	505	0.74	573	0.84	371	0.54	505	0.74	595	0.87	361	0.53	531	0.78
other MOTs	8,692	12,075	1.39	9,010	1.04	9,646	1.11	9,227	1.06	9,220	1.06	9,685	1.11	9,337	1.07
total	9,866	12,919	1.31	10,141	1.03	10,365	1.05	10,222	1.04	10,408	1.06	10,429	1.06	10,382	1.05



(a) u = 20

(b) u = 150

(c) u = 300

Figure 2.8: Total cost split into cost of MOTs and cost of car-type1 (=combustion engine cars) for $u = 20, 150, 300$ and $m = 40$ and different objective function for *VShP-1T:car*. OF:base shows the result with the previously introduced objective function, OF:time only considers the time part. Note that we solve the models with the different objective functions, but afterwards calculate the total cost to make them comparable.

2.3.6 Comparing objective functions

In the following, we compare two objective functions: (1) we take the objective function as presented above, consisting of operational distance cost including cost of time (OF: base), and (2) only incorporating the time factor (OF: time). Again, we show the results for both *VShP-1T:car* and *VShP-xT* as our base cases. With this we aim to see the main driver of our outputs. Note that for the following results we solve the models with the different objective functions, but afterwards calculate the total cost to make them comparable.

Figure 2.8 shows the composition of the total cost for *VShP-1T:car* having a number of users $u = 20, 150, 300$ and cars $m = 40$. We show the cost share of cars and other MOTs. In all three cases we observe a higher cost of the cars when using OF:time as the objective. However, the total cost is only slightly higher for OF:time, resulting in less cost for other MOTs when taking time components as the objective only.

In Table 2.9 we confirm the above figures with numbers. The table is decomposed into results for OF:base, OF:time and the comparison of the two, where we assume (OF:time/OF:base). The first two are given in absolute numbers, the latter as a ratio of the two. Each partition gives the results of the combustion engine cars (car-type1), other MOTs and in total. The numbers are given on average over all instances and all sizes of m. We can see, that using time only as an objective function gives slightly higher overall cost. The smallest difference can be observed for $u = 50, 100$, and ranging around 1.01-1.04 times the cost for all instances. Moreover, we can see that this difference is mainly driven by the higher cost of cars for most of the cases. The OF:time has higher

car cost for all the stated averages in Table 2.9. More detailed information for different sizes of the car fleet (m) can be found in Table A8 in the Appendix A.1.

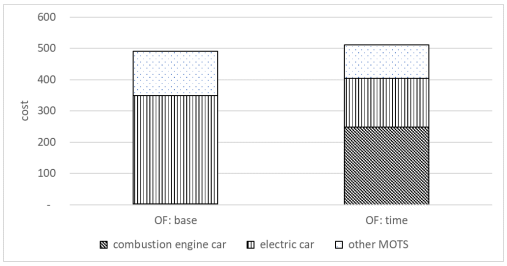
Table 2.9: Total cost for OF:base and OF:time and their comparison calculated as (OF:time/OF:base), split into cost of MOTs and cost of car-type1 (=combustion engine cars) for an increasing u and averaged over all m. OF:base shows the result for the previously introduced objective function, OF:time only considers the time part. Note we solve the models with the different objective functions, but afterwards calculate the total cost to make them comparable.

	u	20	50	100	150	200	250	300
OF:time	car-type1	333	747	1,004	1,089	1,185	1,283	1,277
	other MOTs	208	873	2,267	4,270	6,100	8,130	8,871
	total	541	1,620	3,271	5,359	7,285	9,413	10,147
OF:base	car-type1	258	656	955	1,052	1,148	1,255	1,227
	other MOTs	275	942	2,281	5,917	4,132	7,767	8,704
	total	533	1,599	3,235	5,184	7,065	9,022	9,932
OF:time / OF:base	car-type1	1.29	1.14	1.05	1.04	1.03	1.02	1.04
	other MOTs	0.76	0.93	0.99	1.03	1.03	1.05	1.02
	total	1.02	1.01	1.01	1.03	1.03	1.04	1.02

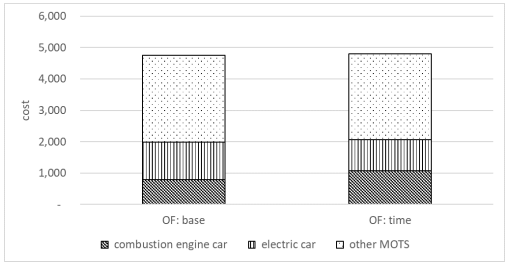
Figure 2.9 plots the composition of the total cost for $u = 20, 150, 300$ and $m = 40$ solving *VShP-xT* and different objective functions. The costs are divided into cost for combustion engine cars (car-type1), electric cars (car-type2) and other MOTs. Again, having the actual objective function, leads to lower overall cost and lower cost of cars. Note that for the time function there is no difference between the two car types, as the differences are only in the operational cost. As previously, the *VShP-xT* prefers electric vehicles, as they have lower cost not related to time.

Table 2.10 summarizes the cost for OF:base and OF:time, and gives the respective ratio of them, calculated as (OF:time/OF:base), for *VShP-xT*. We again give the respective values for the car types, other MOTs and in total. We observe, that if only optimizing towards savings in time, we end up with higher overall cost. We have between 1.01 and 1.03 times the cost compared to our basic objective function. Results separated for each $m = 4, 8, 20, 40$ can be found in Table A9 in the Appendix A.1.

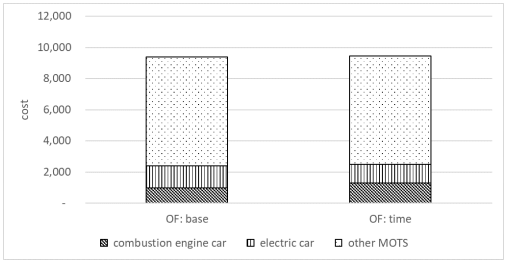
Finally, we compare the average number of trips per car in Tables 2.11 and 2.12 for *VShP-1T:car* and *VShP-xT*, solving each with the different objective functions. For *VShP-1T:car* we observe an increase in trips per car, where we have 1.1 more trips on average for all user groups (u). For *VShP-xT* we detect a different picture. We can see a bigger increase in average trips per combustion engine car, and a decrease of trips for electric vehicles. This is because when considering OF:time the two vehicle types do not make any difference. The difference between the two only concerns the operational cost in this case.



(a) u = 20



(b) u = 150



(c) u = 300

Figure 2.9: Total cost split into cost of MOTs and cost of car-type1 and car-type2 (= combustion engine and electric cars) for $u = 20, 150, 300$ and $m = 40$ and different objective functions for *VShP-xT*. OF:base shows the results for the previously introduced objective function, OF:time only considers the time part. Note we solve the models with the different objective functions, but afterwards calculate the total cost to make them comparable.

Table 2.10: Total cost for OF:base and OF:time and their comparison calculated as (OF:time/OF:base), split into cost of MOTs and cost of car-type1 and car-type2 (= combustion engine and electric cars) for an increasing u and averaged over all m. OF:base shows the results with the previously introduced objective function, OF:time only considers the time part. Note that we solve the models with the different objective functions, but afterwards calculate the total cost to make them comparable.

	u	20	50	100	150	200	250	300
OF:time	car-type1	177	390	507	539	603	650	636
	car-type2	141	326	455	503	533	577	585
	other MOTs	211	874	2,267	4,270	6,099	8,130	8,871
	total	528	1,590	3,229	5,312	7,234	9,358	10,092
OF:base	car-type1	50	193	365	417	463	496	491
	car-type2	225	446	545	589	638	704	683
	other MOTs	239	919	2,273	4,121	7,003	7,754	8,692
	total	514	1,558	3,184	5,127	7,003	8,954	9,866
OF:time / OF:base	car-type1	3.51	2.02	1.39	1.29	1.30	1.31	1.30
	car-type2	0.63	0.73	0.83	0.85	0.83	0.82	0.86
	other MOTs	0.88	0.95	1.00	1.04	0.87	1.05	1.02
	total	1.03	1.02	1.01	1.04	1.03	1.05	1.02

Table 2.11: Average number of trips per car when solving OF:base and OF:time and their comparison stated as OF:time / OF:base for an increasing u and averaged over all m for *VShP-1T:car*.

$u =$	20	50	100	150	200	250	300
OF:time	1.6	1.9	2.1	2.2	2.3	2.4	2.3
OF:base	1.4	1.8	2.0	2.0	2.0	2.2	2.1
OF:time / OF:base	1.1	1.1	1.1	1.1	1.1	1.1	1.1

Table 2.12: Average number of trips per car when solving OF:base and OF:time and their comparison stated as OF:time / OF:base for an increasing u and averaged over all m for *VShP-xT*.

$u =$		20	50	100	150	200	250	300
OF:time	car-type1	1.5	1.9	2.1	2.2	2.3	2.5	2.3
	car-type2	1.5	1.9	2.1	2.2	2.3	2.4	2.3
OF:base	car-type1	0.8	1.3	1.5	1.5	1.6	1.7	1.6
	car-type2	1.5	2.1	2.4	2.4	2.5	2.7	2.6
OF:time / OF:base	car-type1	1.8	1.5	1.4	1.4	1.4	1.4	1.4
	car-type2	1.0	0.9	0.9	0.9	0.9	0.9	0.9

2.3.7 Managerial implications and discussion

We have seen in all our results, that with a higher number of cars (combustion engine or electric car), we enforce lower total cost. This is true, even though cars are the most expensive MOT from an operational cost point of view. However, they are in many cases fast MOTs. Therefore, if possible, the trips are covered by a car. Also, whenever possible, electric vehicles are preferred as they have even lower cost but the same speed as the conventional ones. This holds for the case where only one kind of car is shared. A mixed fleet is slightly more expensive compared to when only electric vehicles are employed. However, if a fleet of combustion engine cars is available, one can gradually expand the fleet or replace the conventional cars with the electric vehicles. Having a shared pool of only combustion engine vehicles is the most expensive case of the discussed sharing concepts and the least environmental friendly, and thus not recommendable.

Employing no cars at all, is most expensive. If one decides to go with any (of the presented) sharing concepts, it is advisable that the number of cars in the fleet should be at least 20%-25% of the number of users. E.g. for 20 users this would be 4-5 cars. From there it starts to be cost efficient to have shared vehicles, and additionally cover trips with other MOTs such as public transportation or bike. The use of electric vehicles - either exclusively or in combination with conventional cars - is highly recommendable due to their lower operational cost and same time needed for trips.

Using operational cost information and time in the objective function is crucial. As the cost of time depends on the distance too (we assume different distances for different MOTs), not all of the trips are covered by the fastest MOT, which would be a car. So the shortcuts that can be taken by different MOTs sometimes outperform the benefits

given by fast cars. As our instance companies are based on a city, this makes sense. For longer trips, the results would lead to different trade-offs.

If enough cars are available to cover the beneficial trips without handing over the cars at the depots, then this will be done so and is also recommendable. A profound sharing concept is only advisable if the car is a restricted resource (however, not too much as discussed above). Yet, we saw in our results, that with a constant number of users but a smaller fleet, the trips per car are rising above the average number of trips a user is taking. Also, the average number of trips per car is higher for electric vehicles.

Finally, we introduced a set of restricted MOTs based on individual user preferences. As expected, the case where all MOTs are always available for all trips, and thus for all users, is the one with the least cost as it is the least restricted case. However, for some of the cases we observed only a modest increase in cost. Yet, by giving the users a restricted set of MOTs we might achieve a higher satisfaction and acceptance of the system and therefore it can be beneficial in a non-monetary way.

2.4 Conclusion

Inspired by the change of mobility and vehicle-sharing systems we proposed two modeling approaches for a vehicle-sharing problem. In our problem we assume a set of users that have to cover certain trips on a fixed time schedule. These trips are then covered by a certain mode of transport. We assume a restricted available set of shared vehicles, e.g. a pool of cars, which the users may use. Other modes of transport are incorporated without any capacity limits. We aim to assign the restricted resources in the best possible way such that savings (using e.g., a car instead of any other mobility type) are maximized. Note that our initial framework considers a sharing system within a company, however the models can be applied to any community with a closed group of users.

We used two well-known formulations from the literature, namely the maximization equivalent of the minimum-cost flow problem and the multi-commodity flow problem. If we assume only one shared MOT, e.g. cars, we base our formulation on the minimum-cost flow problem. We extend the problem by introducing another type of shared vehicle, and we formulate it as a multi-commodity flow problem where the commodities are the shared vehicles. Note that a shared resource may also be a bike or another MOT.

We further provide managerial insights for a company incorporating combustion engine vehicles and electric cars as our shared vehicles. We show that a shared fleet of electric vehicles contributes most to our objective function. Instances with up to 300 users are solved in less than 20 seconds of computing time. With this we can show that our models can be used on a daily operational basis.

Besides the analysis, the present paper aims to give a theoretical foundation to future vehicle-sharing problems. As the models are well studied in the literature, many efficient algorithms exist and even bigger instances can be solved to optimality within seconds. Future work might look into adapting the structure of the trips. Now we assume a fixed sequence, however optimizing the trips as a small-sized traveling salesman problem may achieve even better results.

Acknowledgements

This work was supported by the Climate and Energy Funds (KliEn) [grant number 853767]; and the Austrian Science Fund (FWF): P 31366.

Chapter 3

Modeling and solving the multimodal car- and ride-sharing problem

Published in: *European Journal of Operational Research*
Modeling and solving the multimodal car- and ride-sharing problem.
M. Enzi, S.N. Parragh, D. Pisinger & M. Prandtstetter.
https://doi.org/10.1016/j.ejor.2020.11.046

Abstract We introduce the multimodal car- and ride-sharing problem (MMCRP), in which a pool of cars is used to cover a set of ride requests while uncovered requests are assigned to other modes of transport (MOT). A car's route consists of one or more trips. Each trip must have a specific but non-predetermined driver, start in a depot and finish in a (possibly different) depot. Ride-sharing between users is allowed, even when two rides do not have the same origin and/or destination. A user has always the option of using other modes of transport according to an individual list of preferences.

The problem can be formulated as a vehicle scheduling problem. In order to solve the problem, an auxiliary graph is constructed in which each trip starting and ending in a depot, and covering possible ride-shares, is modeled as an arc in a time-space graph. We propose a two-layer decomposition algorithm based on column generation, where the master problem ensures that each request can only be covered at most once, and the pricing problem generates new promising routes by solving a kind of shortest-path problem in a time-space network. Computational experiments based on realistic instances are reported. The benchmark instances are based on demographic, spatial, and economic data of Vienna, Austria. We solve large instances with the column generation based approach to near optimality in reasonable time, and we further investigate various exact and heuristic pricing schemes.

Keywords— transportation, car-sharing, ride-sharing, vehicle scheduling problem, column generation

3.1 Introduction

Studying the development of mobility during the last decades, one can easily observe that we are facing a strong wind of change. While some years ago the main developments

were related to technological improvements, the introduction of e-mobility was a first step towards an ongoing kind of revolution. From there, a new understanding of mobility developed and we are now facing a future where "owning cars" is replaced by "being mobile". Users preferably only specify cornerstones of their travel — such as origins and destinations, latest arrival times or preferable modes of transport (MOT) — and rely on an information system to provide an (optimal) assignment of modes of transport to their demands. This attitude is supported by mobility concepts like vehicle sharing (car and bike), easy access to mobility via mobility cards, and Mobility as a Service [113]. We observe these developments not only in the private sector but also in the area of corporate mobility. Increasingly, companies are trying to change their view on their corporate mobility by switching from individually assigned cars towards Mobility as a Service for their employees. Companies strive to have an overall green and sustainable profile and employees are aware of the importance to contribute to a greener world, even if their travel time of a trip might increase. Instead of supporting further developments in corporate mobility privileging a few selected users, we are aiming at providing sustainable corporate mobility concepts that ensure at least the same level of mobility, while increasing positive impacts (e.g., cost reduction, ecological sustainability, and employee satisfaction).

This work is part of an applied research project *SEAMLESS* (http://www.seamless-project.at), in which the project partners are implementing the discussed ideas including the supporting algorithms in their companies. The major goal of the project is the development of novel corporate mobility concepts aiming at providing mobility to the company (and its employees) instead of only providing cars. This includes the introduction of car pools that can be used by the employees on a smart assignment strategy. Additional modes of transport are incorporated like bikes, (public) bike- and car-sharing, public transport and users can co-ride with each other. Ride-sharing saves resources, such as cars and energy, it is considered to have a good environmental footprint and can solve congestion problems. Masoud and Jayakrishnan [107] report that for private cars in the US with four seats, only around 1.7 seats are actually used on average. This number decreases to only 1.2 for work-based trips, which shows the underutilization of cars, especially company cars. The increasing number of empty seats in cars and an increasing number of users asking for rides, imply motivation to elaborate a sophisticated ride-sharing system. Furthermore, not only the sharing economy is increasing but also a combined and integrated use of various modes of transport. To avoid pollution and congestion problems, various cities give incentives to use (a combination of) "greener" modes of transport [133, 46]. As transportation is one of the biggest producers of emissions [65], it is vital to consider sustainability aspects, shift to more sustainable modes and enhance the environmental footprint.

The package of mobility offers can be seen as an extended car pool. It is crucial to assign the right vehicle to the right mobility need — e.g., if someone is aiming to travel a short distance in the city, public transport is better suited than a conventionally driven minivan. In return, the minivan is the right choice if an employee has to transport some special equipment to a meeting at a location about 300 kilometers away. This implies that it is necessary to estimate the mobility demand, allowing the user to specify preferred modes of transport, and to determine the number (and types) of cars to be owned in the car pool as well as the mobility offers provided to the employees like mobility cards or access to public vehicle sharing systems. Although sharing reduces

costs and environmental impact, the complexity in fleet management increases. This directly implies that computational support is necessary to be able to efficiently handle the fleet.

We study the multimodal car- and ride-sharing problem in a company having one or more offices from where the employees have to visit various customers during office hours (e.g., for business meetings). We consider a fixed and unique employee-to-meeting assignment and a fixed latest arrival time. This results in a fixed sequence of tasks (also referred to as trip) for every employee with several stops, starting and ending at predefined (but possibly different) depots. A pool of vehicles is provided to the employees (also referred to as users) who can jointly use them (car-sharing). Furthermore, up to two users may co-ride on specific legs or routes with each other (ride-sharing).

We model the trips as arcs in a directed acyclic graph. Vehicle routes consist of one or more trips, whereas the driver of these trips may change at the depot. Similar to the vehicle scheduling problem, the available vehicles cover the scheduled trips resulting in vehicle routes. As the pool of cars is restricted, only a subset of the trips will be covered by the shared vehicles. In order to cover all mobility requests in the best possible way, further MOTs such as bikes or public transport are used. If a trip is not covered by car, the cheapest other MOT will be used.

This paper and the project objectives focus on adapting future mobility considerations to a corporate setting. However, the results can easily be adapted for different closed groups with a predefined set of users, such as home communities, suburban areas, or simply a network of users with predefined locations where the cars must be picked-up at and returned to. Furthermore, the model can easily be adapted to bikes, segways, cargobikes, (electric) scooters, and other sharing offers.

As the problem is modeled as an extended vehicle scheduling problem (VSP) with multiple depots, we contribute to the body of this specific problem too. The VSP assigns a set of vehicles to a set of scheduled trips, such that costs are minimized and each trip is covered by exactly one vehicle [14]. There are three main differences between the vehicle scheduling problem and the MMCRP. First, in our case not all trips need to be covered. Second, the multi-depot VSP assumes that all vehicles return to their original depot. We allow for different, but predetermined, start and end points. Third, we have to avoid that users co-ride in parallel on different trips. Hence we add a tailored ride-sharing constraint to the model. The MMCRP can be transformed into a kind of vehicle scheduling problem by having infinite cost for all other modes of transport, forcing the solution to cover all trips and allowing for different start and end depots. The MMCRP can also be formulated as a VSP with profits, which only - related to the idea of the vehicle routing problem (VRP) with profits - covers trips that are profitable.

The contributions of this paper are as follows:

- We introduce the novel MMCRP derived from a real-world application, and formulate it as an extended vehicle scheduling problem. To the best of our knowledge, it is one of the first models including both car- and ride-sharing. We show that this real-world application can be efficiently solved with well-known methods.
- We present a two-layer decomposition of the problem. In the first layer, trips starting and ending at a depot are enumerated. The trips also take care of enumerating all possible ride-sharing possibilities. With this we are able to hide complicated

constraints considering ride-sharing and make it usable for practical purpose. The framework is flexible and allows the introduction of additional constraints like detour constraints, co-riding preferences, and driving time constraints. In the second layer, the trips are combined into vehicle-routes. The second-layer decomposition is solved through a column generation based algorithm. We present an efficient algorithm for solving the pricing problem using a label setting algorithm on a directed acyclic graph (DAG). Several different pricing strategies are presented and compared. These include adding more columns in each iteration, and various heuristics in combination with an exact approach for solving the pricing problem.
- Computational results confirm that large instances can be solved to near-optimality in reasonable time using a column generation based approach, making it possible to use the algorithm for daily planning of multimodal car- and ride-sharing systems. We also show that the gap between the LP-bound found through column generation and the integer solution obtained on the same columns is very small.

The paper is organized as follows: First, in Section 3.2, we discuss related work focusing on car- and ride-sharing as well as the VSP. Then, we provide a detailed problem description of the MMCRP in Section 3.3. The solution approach and the auxiliary graph that is used in our algorithm are described in Section 3.4. Based on the auxiliary graph we present a direct formulation of the MMCRP in Section 3.4.1. In Section 3.4.2, we present a path formulation of the problem and show how its linear relaxation can be solved through delayed column generation in Section 3.4.3. In Section 3.4.4 we explain how the pricing problem can be solved through dynamic programming and propose various heuristics for improving the computational effort. Section 3.5 presents the computational experiments. The paper is concluded in Section 3.6 by summing up the achieved results, and proposing ideas for future research.

3.2 Related work

Recently, car- and ride-sharing have received considerable attention, and several variants of the problem have been studied. Mourad et al. [111] provide a thorough overview on models and algorithms for optimizing shared mobility. In the following section, we review closely related problems, including car-sharing, ride-sharing, and the vehicle scheduling problem focusing on column generation based approaches.

3.2.1 Car-sharing

Car-sharing systems involve a pool of cars that are shared among a set of users, who are usually known in advance (in public car-sharing systems these would be subscribers, in our case employees). The MMCRP without ride-sharing reduces to a car-sharing problem. Jorge and Correia [88] and Brandstätter et al. [34] review car-sharing optimization problems in detail. Most optimization studies consider publicly available systems and focus on rather strategic problems. In our setting we consider a car-sharing system available to a closed community only and focus on planning the daily operations. Many studies focus on public car-sharing systems and tackle problems such as charging station placement [32, 35] or relocation of cars between stations [91, 32]. Latest works increasingly

also tackle the operational characteristics of car-sharing systems such as the effect of temporal and spatial flexibility on the performance of one-way electric car-sharing systems [31], integrating pickups and deliveries on shared vehicle routes [23], or dynamic relocation policies [123].

3.2.2 Ride-sharing

Ride-sharing describes co-riding of one or more users between an origin and a destination or sub-paths of it. This is also the main idea of the MMCRP, where we do not only exploit the various MOTs in the best possible way, but try to merge rides by allowing ride-sharing if it is beneficial. In the following we review some related studies addressing ride-sharing.

Dial-a-ride problems (DARP) or the closely related pick-up-and delivery problem (PDP) are often used to formulate ride-sharing activities [85, 99]. Related to the MMCRP is also PDP with transfer [108, 121, 47]. An exhaustive review of these problems can be found in Ho et al. [84].

Masoud and Jayakrishnan [107] propose a decomposition algorithm to solve a many-to-many ride-matching problem to optimality in a time-expanded network. Participants only provide the origin, destination and latest/earliest times, which is similar to our problem statement. In contrast to our problem, they strictly split riders and drivers. Huang et al. [86] formulate a two-stage problem minimizing total cost for long-term car-pooling. Drivers are selected, passengers assigned, and for each driver a traveling salesman problem (TSP) is solved considering constraints regarding fairness and preferences. Bit-Monnot et al. [26] compute a driver's and passenger's individual paths including the mutual sub-path between two (to be determined and synchronized) points. Mutual trips are followed by their individual paths towards the driver's and passenger's destination. As in our work, they also include public transport and walking before/after ride-sharing, however the focus of the work is to determine the optimal pick-up and drop-off locations for requests.

A number of works study commuter trips [15, 94, 122], whereas we focus on trips during working hours from/to meetings with customers. Chen et al. [44] aim at minimizing the cost of commuters and business traffic of a company, which consists of the cost incurred from vehicle miles and the costs of penalizing the efficiency losses (arriving too late at meetings, waiting time for transfers, inconvenience and risk with transfers). A constructive heuristic based on savings in miles driven and cars used is introduced. The problem definition is closely related to ours, as not only commuting trips are considered but also business traffic, i.e., travels between meetings. Moreover, they also employ savings as a objective but only use a heuristic approach to solve the problem.

Contrary to other papers, we model our compact problem as a kind of vehicle scheduling problem defined on an acyclic time-space graph, where we do not model pick-ups and deliveries explicitly, but enumerate all possible ride-shares in an auxiliary graph which is used as input to the second stage model.

3.2.3 Vehicle scheduling problem

The VSP received increasing attention in the early 80s [28, 29], and is mainly applied to time-tabled trips of public transport or crew scheduling. In the following we give a

short overview on recent works on the multi-depot variant of the problem (MDVSP) using column generation based approaches to solve it. Further works elaborate the idea of the MDVSP by introducing alternative-fuel vehicles [3] or considering a heterogeneous fleet of vehicles [79]. An overview of basic vehicle scheduling models is given in Bunte and Kliewer [39]. The MMCRP can be seen as a MDVSP with profits. The literature on the VRP with profits or closely related Orienteering Problem is vast [130, 81]. We could not find any publications on the VSP with profits.

The MDVSP was proven to be NP-hard by Bertossi et al. [24]. Column generation was first applied to the MDVSP by Ribeiro and Soumis [125] and extended by Hadjar et al. [82] and Groiez et al. [78]. Note that these algorithms focus on proven integer optimality of the entire problem, which is not the case in our work. Pepin et al. [120] compare five heuristic for the MDVSP and conclude that the column generation heuristic performs best assuming enough computational time is available and stability is required. Guedes et al. [80] propose a simple and efficient heuristic approach for the MDVSP. The heuristic first applies state space reductions to reduce complexity and thereafter a truncated column generation approach. Kulkarni et al. [97] present a new inventory formulation for the MDVSP and a column generation based heuristic proposing a novel decomposition. The multi-depot vehicle scheduling with controlled trip shifting [54] is closely related to the MMCRP. The generalization of the MDVSP allows for slight modification of one trip scheduled time. Trips are multiplied, representing each trip for different starting times. The aim is to find a set of bus schedules that covers every trip exactly once by satisfying vehicle availability and minimize costs. The work introduces a two-phase matheuristic where column generation solving the linear relaxation is embedded in a diving heuristic to derive an integer solution. The sequence of trips is fixed in the first phase by the column generation approach and thereafter the copies of a trip are chosen using a mixed integer program.

Note that all of the above works use either variable fixing or rounding strategies in their approaches. We solve the linear relaxation to optimality by column generation and find the integer solution by solving the original model using the obtained columns. Furthermore, we do not tackle the standard VSP but a kind of MDVSP with profits in which only beneficial arcs are covered by a vehicle. The study by Oukil et al. [117] is structurally similar to ours, but having some important differences. Oukil et al. [117] focus on the comparison of the standard and the stabilized column generation approach, and discuss the impact of different time horizons. Both, the stabilization of the column generation and different time horizons, are not subject to our study. They emphasize on the methodological contribution. We combine a theoretical and practical contribution and apply the standard column generation approach, extended by heuristic approaches, to solve the LP-relaxation. Moreover, the underlying graphs differ. As we model ride-sharing directly into the graph we have multiple possibilities to cover tasks and trips. Therefore, we have to make sure that a user is not driving in parallel. Moreover, we allow the vehicle routes to start and end in different depots, which is not the case in Oukil et al. [117]. Similar to Desfontaines and Desaulniers [54], we work on a multi-graph in which copies of links represent the same connection at different times and, in our case also involving different ride-sharing activities.

3.3 Problem description

We study the multimodal car- and ride-sharing problem in a company having one or more offices from where the employees have to visit various customers during office hours (e.g., for business meetings). Assuming that a company operates different offices (also referred to as depots), an employee might work in any depot. We note that even though the cases where employees switch their work place are rather rare, we included them because they were mentioned by our company partners. Therefore, it is not necessary that the employee returns to the starting depot after her meeting with a customer. Each customer visit involves one specific employee. We consider fixed and unique employee-to-meeting assignments and a fixed latest arrival time. As we consider business meetings at the customer locations, we assume that even if the employee arrives earlier, the starting time of the meetings will not change. Knowing the fixed starting time of the meeting as well as the length of it, we can calculate in advance the earliest departure time of each ride and do not have to explicitly consider the time of the meeting. This results in a fixed sequence of tasks for every employee with several stops, starting and ending at predefined (but possibly different) depots. We call such a fixed sequence of nodes a trip.

The company operates a finite number of vehicles at each depot and provides possibilities to use other modes of transport such as public transport, bikes, taxis or walk. We assume no start-up cost is associated with vehicles, and depots must have a specific number of vehicles at the beginning and end of the day. With this we assume that we do not have to account for relocations of cars. The employee specifies which modes can be used, since e.g., a person without a driving license cannot be the driver of a car. Moreover, the cars are only interchanged at the depots. This restriction is given from the project partners as changing cars at customer locations would imply too much inconvenience and they reported limited acceptance for handing over cars during a trip. For example, as the meetings of different users are usually not at the same location and/or time, one would need additional meeting points and/or times for the hand-over of the car. Further we consider ride-sharing, which is allowed between users, even when two rides do not have the same origin and/or destination. Ride-sharing may at most involve one co-rider. For further details on users, trips and MOTs see Section 3.3.1. Ride-sharing is described in more detail in Section 3.3.2.

The MMCRP aims at determining the optimal MOT-assignment for each trip and to schedule the routes of the cars, maximizing savings when using a car including ride-sharing compared to any other mobility type whilst ensuring that all customers are visited at the right time by the right employee. The cost for the savings calculations does not only include distance cost but also cost of time (as hourly wages of employees) in order to properly reflect the trade-off between fast (but expensive) and slow (but cheap) MOTs. The savings calculation is outlined in Section 3.3.3. A vehicle route depicts a route of a vehicle during the day encompassing one or more drivers, handing over the vehicle at a depot including possible ride-sharing activities. Note that for our problem it is sufficient to only explicitly model car routes, as we only schedule the trips for the limited resource (i.e., cars). The remaining trips are assigned to the cheapest other MOT a user is willing to choose. This relies on the realistic assumption that users will rationally choose the next cheapest possibility to travel, if a car is not available. For a better understanding of the problem, an illustrative example is given in Section 3.3.4.

3.3.1 Users, trips and modes of transport

We have u users and n tasks given. Each user p has a sequence of tasks $Q_p = (q_p^1, q_p^2, \ldots, q_p^{n_p})$ that need to be covered. User p starts at depot a_p and finishes at a (possibly different) depot b_p according to the user's wishes. A trip π denotes the sequence of nodes of user p starting at a_p and ending at b_p. Each task q_p^i is associated with a latest arrival time and earliest departure time. As we assume a fixed starting time of the task (i.e., the latest arrival time) we also know the earliest departure time by adding the duration of the task to it. Doing so, we do not have to explicitly consider the duration. The driving time between two tasks (q_p^i, q_p^j) using mode of transport k is $t^k_{q_p^i, q_p^j}$, while the cost is $c^k_{q_p^i, q_p^j}$. We consider a set of modes of transport $K = \{car, walk, bike, public, taxi\}$. Every user p has a set $K^p \subseteq K$ of possible modes of transport that can be used. We assume that a pool W_d of shared cars is available at depot $d \in D$ at the beginning of the day, and that $\overline{W}_d$ cars should be returned to the depot at the end of the day. Depots $d \in D$ reflect the depots where the cars are parked and the trips start and end. Note that all start and end nodes of a trip a_p, b_p are connected to the depots d. The demand at the end of a day will typically reflect the forecasted cars needed at the depot on the following day. For the other mobility types (car, walk, bike, taxi), we assume that there is infinite capacity.

If a trip π is started by a car, then the car should be used for the full trip. However, ride-sharing can take place between any two nodes of a trip driven by the car. If the co-riding user does not follow the driver for the full trip, then we assume that the cheapest other MOT is used for the rest of the trip. We assume that if a user does not use a car on her own or is not co-riding, then she will take the cheapest other MOT included in her set of MOTs in order to conduct her trip.

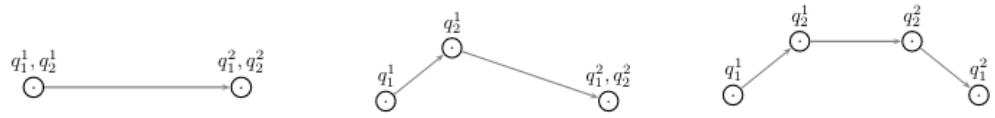
3.3.2 Ride-sharing

Employees can share a ride if it is beneficial. Usually this applies if meetings are visited together or different meetings are nearby or lie on the colleague's trip. We distinguish between three ride-sharing types: (1) co-riding users share the same origin and destination, (2) they have the same origin and distinct destinations or vice versa, (3) they have different origins and destinations. In the following some representative examples are provided.

Each user p has to cover a set of tasks $Q_p = (q_p^1, q_p^2, \ldots, q_p^{n_p})$. We are considering ride-sharing between two users $p = 1$ and $p = 2$ on a leg between two tasks (q_p^i, q_p^j), thus going from q_1^1 to q_1^2 for user $p = 1$ and another leg going from q_2^1 to q_2^2 for user $p = 2$. Although our framework can easily be generalized to multi-user ride-sharing, we only consider two user ride-sharing to ensure user satisfaction. By allowing multiple users to share a ride, a user might end up using a disproportional part of the time as a driver for others.

The simplest ride-sharing case occurs when users $p = 1$ and $p = 2$ have the same origin and destination, as shown in Figure 3.1(a). In this case both users can be served on the ride.

We can also have the case in which only the source or destination is shared. Starting with the case where the end destination is shared we have the case shown in Figure



(a) Two rides with same origin and destination in time-space representation.

(b) Two rides with shared destination in time-space representation.

(c) Two rides with distinct origin and destination in time-space representation.

Figure 3.1: Three examples of ride-sharing illustrated in a time-space network. User $p = 1$ is going from q_1^1 to q_1^2 while user $p = 2$ is going from q_2^1 to q_2^2.

3.1(b). In this case, user $p = 1$ has to drive from q_1^1 to q_2^1 to pick up user $p = 2$, and then both drive to the shared end destination where $q_1^2 = q_2^2$. Provided that all time limits are satisfied, the cost of the ride can be calculated as the sum of the individual legs. The case where the origin is shared is handled in a symmetric way. Obviously, this shared ride is only beneficial if the detour for $p = 1$ is not too large.

In general, both origin and destination can be distinct as illustrated in Figure 3.1(c). In this case user $p = 1$ has to drive from q_1^1 to q_2^1 to pick up user $p = 2$, drive this user to her destination q_2^2 and then drive to her own destination q_1^2. Provided that all time limits are satisfied, the cost of the ride can be calculated as the sum of the individual legs. Please note that the end destination of the driver (q_1^2) must always lie after an intermediate point (e.g., q_2^2) as we do not allow to change drivers on the trip.

3.3.3 Savings calculation

We calculate cost and travel time for each trip. In order to reach the best choice of MOT combination we aim to obtain savings when using a car including ride-sharing compared to any other mobility type rather than focusing on minimizing costs only. Saving γ_π of a trip π is the sum over all savings $\gamma_{q_p^i}$ of included tasks q on trip π. We consider costs of subsequent tasks (q_p^i, q_p^j) and obtain the savings calculation by considering detouring for ride-sharing. Note that the obtained savings might also be negative. This will occur if the cheapest MOT for a trip is not the car.

In a first step let us compute the savings obtained when no ride-sharing between two subsequent tasks (q_p^i, q_p^j) is considered. The savings of tasks q_p^i and its fixed successor q_p^j can be calculated as the difference between cost of using the cheapest other MOT $c^k_{(q_p^i, q_p^j)}$ and cost of using the car $c^{car}_{(q_p^i, q_p^j)}$, such that:

$$\gamma_{q_p^i} = \min_{k \in K \setminus \{car\}} \{c^k_{(q_p^i, q_p^j)} - c^{car}_{(q_p^i, q_p^j)}\} \tag{3.1}$$

Next, let us assume two users $p = 1$ and $p = 2$ whereas ride-sharing is demanded between two tasks (i, j) of user $p = 2$, (q_2^i, q_2^j). We add detour costs to go to/from these tasks. We have to account for additional costs of going from the driver's ($p = 1$) task q_1^i to the starting point of the demanded ride-sharing q_2^i as well as additional cost from the ride-sharing drop-off point q_2^j to the driver's original task q_1^j. This gives us an additional detouring cost between (q_1^i, q_2^i) as well as (q_2^j, q_1^j) for which we only take into account the cost of using the car. We do not change the fixed sequence of a user's trip, however, we

must keep track of whether ride-sharing is conducted between two trips in order to take into account additional detouring cost. Hence for not traversing the original link (q_1^i, q_1^j) we save $c^k_{(q_1^i,q_1^j)}$ and additionally save costs by allowing for ride-sharing between $c^k_{(q_2^i,q_2^j)}$. Therefore, for each task q_1^i including subsequent ride-sharing we compute the savings $\gamma_{q_1^i}$ as follows:

$$\gamma_{q_1^i} = \min_{k \in K \setminus \{car\}} \{(c^k_{(q_1^i,q_1^j)} + c^k_{(q_2^i,q_2^j)}) - (c^{car}_{(q_2^i,q_2^j)} + c^{car}_{(q_1^i,q_2^i)} + c^{car}_{(q_2^j,q_1^j)})\} \tag{3.2}$$

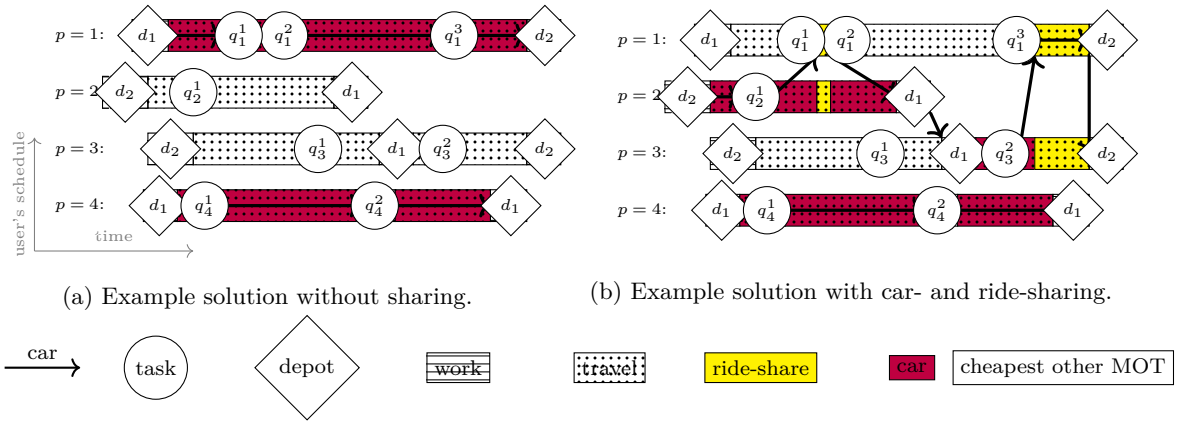
assuming q_1^j is its fixed successor and ride-sharing is employed for tasks (q_2^i, q_2^j) between the original sequence (q_1^i, q_1^j). The saving of trip π is then calculated as $\gamma_\pi = \sum_{q \in \pi} \gamma_q$.

In order to provide a more understandable overview, let us assume a trip π of user $p = 1$ considering three tasks to be visited, q_1^1, q_1^2 and q_1^3. The trip starts at a_1 and ends at b_1. The user's trip would then be a_1 - q_1^1 - q_1^2 - q_1^3 - b_1. The corresponding savings calculation for trip π and without considering ride-sharing is as follows: $\gamma_\pi = \min_{k \in K \setminus \{car\}} \{(c^k_{(a_1,q_1^1)} - c^{car}_{(a_1,q_1^1)}) + (c^k_{(q_1^1,q_1^2)} - c^{car}_{(q_1^1,q_1^2)}) + (c^k_{(q_1^2,q_1^3)} - c^{car}_{(q_1^2,q_1^3)}) + (c^k_{(q_1^3,b_1)} - c^{car}_{(q_1^3,b_1)})\}$. In a next step, assume that user $p = 1$ takes a detour between q_1^2 and q_1^3 in order to ride-share with user $p = 2$ between q_2^1 and q_2^2. Now the sequence would be a_1 - q_1^1 - q_1^2 - q_2^1 - q_2^2 - q_1^3 - b_1 whereas the fixed successors for our calculations do not change, as described above. We get the respective savings of the above presented trip π: $\gamma_\pi = \min_{k \in K \setminus \{car\}} \{(c^k_{(a_1,q_1^1)} - c^{car}_{(a_1,q_1^1)}) + (c^k_{(q_1^1,q_1^2)} - c^{car}_{(q_1^1,q_1^2)}) + (c^k_{(q_1^2,q_1^3)} + c^k_{(q_2^1,q_2^2)} - c^{car}_{(q_2^1,q_2^2)} - c^{car}_{(q_1^2,q_2^1)} - c^{car}_{(q_2^2,q_1^3)}) + (c^k_{(q_1^3,b_1)} - c^{car}_{(q_1^3,b_1)})\}$. We make these calculations for every variant of trip π representing all possible ride-sharing trips.

Please note that the same task q_p^i can be on different trips and have different savings as they represent different rides. Moreover, for the cases where the driver p and rider p' share their origin and/or destination we do not account for all detouring cost such that $c^{car}_{(q_p^i,q_{p'}^i)} = 0$ and/or $c^{car}_{(q_{p'}^j,q_p^j)} = 0$, respectively.

3.3.4 Illustrative example

To better illustrate the problem, a possible schedule is shown in Figure 3.2. We have 4 users ($p = 1$,$p = 2$,$p = 3$,$p = 4$), 2 cars, and 2 depots (d_1 and d_2). Each user's schedule is depicted by one horizontal line connecting depots d and meetings q_p^i. User $p = 1$ visits q_1^1,q_1^2 and q_1^3, user $p = 2$ is assigned to q_2^1, user $p = 3$ visits q_3^1 and q_3^2 whilst returning in between to depot d_1 and lastly user $p = 4$ drives to tasks q_4^1 and q_4^2. Background rectangles with lines depict duration of a meeting, dots indicate the user is traveling. If the background is not colored, the user is traveling with the cheapest other MOT, purple denotes travel by car, yellow ride-sharing. The arrows illustrate the traveling of the two cars. Figure 3.2(a) shows a possible solution without sharing, Figure 3.2(b) gives an adapted solution with car- and ride-sharing. User $p = 4$ uses in both figures one car for the whole trip and does not share any rides. In Figure 3.2(a) the second car is used by user $p = 1$ for the whole trip. Differently in Figure 3.2(b) where one of the cars is handed over from user $p = 2$ to user $p = 3$ at depot d_1. Furthermore, both drivers of the car take on user $p = 1$ for some legs of her required trip, shown in yellow. Otherwise, user $p = 1$ uses the cheapest other MOT.



(a) Example solution without sharing.

(b) Example solution with car- and ride-sharing.

Figure 3.2: Examples without sharing as well as car- and ride-sharing. We have two offices (depots) d_1 and d_2 and four users $p = 1$, $p = 2$, $p = 3$ and $p = 4$; tasks are denoted as q_p^i. Background rectangles with lines depict duration of a meeting, dots mean the user is traveling. If the background is not colored, the user is traveling with the cheapest other MOT, purple depicts travel by car, yellow ride-sharing. The arrows illustrate the routes of the two cars.

3.4 Solution approach

In order to simplify the problem, we reformulate it using a time-space network, in which all possible ride-sharing trips are represented by arcs. We introduce a graph $G(V, E)$, where vertices are given by their time-space coordinates (i.e., time and depot).

Figures 3.3(a) and (b) show the first step of the graph transformation. We start by modeling the depots d at the start and end of the planning horizon where all trips are connected to and cars are located, as well as a_p and b_p denoting the start and end points of a trip π, and user tasks $q \in Q_p$ depicted as q_p^i. Solid lines denote trips, dotted ones indicate waiting arcs in the set E' denoting that the car is not moving. Hence, we start by considering all trips of all users starting at a_p and ending at b_p including tasks q through the whole planning horizon, which then start and end at depot d. In Figure 3.3(a) we show the starting point of the graph construction whereas user tasks $q \in Q_p$ are still in the graph. From this graph, we transform all a_p, b_p connections (i.e., all trips) to the arcs as represented in Figure 3.3(b). As can be seen, we do not explicitly consider the tasks in the graph any more but save all relevant information on the arcs. For each user $p \in P$ we enumerate all possible trips from the user's start depot a_p to the user's end depot b_p, including possible ride-sharing as described in Section 3.3. Figure 3.3(c) gives two possible ride-sharing trips. Figure 3.3(d) then shows the extension in the trip-based arc representation of Figure 3.3(a). As can be seen, this gives us multiple arcs between nodes a_p and b_p each depicting a possible way how the trip can be conducted. Due to ride-sharing, the start time of two rides may be different even if they consider the same user p starting at the same depot a. A similar observation holds for the end times of two rides for user p.

Every possible trip $\pi \in R_p$ of user p including any number of co-ride possibilities (including 0) results in a tuple $\{(a_p^\pi, b_p^\pi, g_p^\pi, h_p^\pi, s_p^\pi, \ell_p^\pi)\}$. Here we have that a_p^π is the start depot of the ride trip π, b_p^π is the end depot of trip π, g_p^π is the departure time at start

depot a_p^π, h_p^π is the arrival time at end depot b_p^π, s_p^π is the saving of the ride, and ℓ_p^π is a list of visits covered, both for the driver and possible co-rider(s).

For every user p let $V_p = \{(a_p^\pi, g_p^\pi)\}_{\pi \in R_p} \cup \{(b_p^\pi, h_p^\pi)\}_{\pi \in R_p}$ be the set of nodes associated with p as driver of the car. Let the set of ride arcs be $E_p = \{\Big((a_p^\pi, g_p^\pi), (b_p^\pi, h_p^\pi)\Big)\}_{\pi \in R_p}$. The saving of a ride arc s_p^π.

If we assume that σ is the first possible time, and τ is the last possible time in the planning horizon, then for every depot we construct nodes (d, σ) and (d, τ). The set of nodes can now be defined as

$$V = \{V_p\}_{p \in P} \cup \{(d, \sigma)\}_{d \in D} \cup \{(d, \tau)\}_{d \in D}$$

Finally we need to introduce the set E' of waiting arcs. A waiting arc is inserted between nodes $(d, h'), (d, h'') \in V$ in the graph if they correspond to the same depot d and h'' comes immediately after h' (i.e., no other node (d, h) with $h' < h < h''$ exist). Now, the set of arcs can be defined as

$$E = \{E_p\}_{p \in P} \cup E'$$

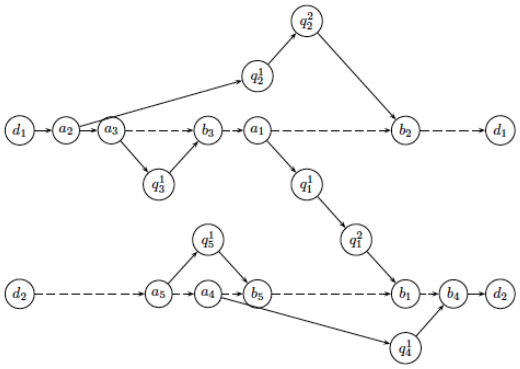
Due to the possibly exponential number of ride-sharing combinations, we can have an exponential number of arcs. However, in practice, the number of possible trips π for each user p is quite limited.

Note that the problem is modeled as a kind of a vehicle scheduling problem with multiple depots. In the vehicle scheduling problem with multiple depots we have a set Z of trips. A trip z has an associated start time l_z and end time l'_z. Two trips z and z' can be run in sequence (i.e., they are compatible) if there is sufficient time to get from z to z'. Moreover, we have multiple depots, each depot d having a capacity W_d. Every trip has to be run by one vehicle, minimizing the driving costs. A major difference between the vehicle scheduling problem and the MMCRP is that in our case not all trips need to be covered. However, the vehicle scheduling problem can either be transformed into the MMCRP by having infinite cost for all other modes of transport, forcing the solution to cover all trips or we it can be seen as an extended vehicle scheduling problem with profits.

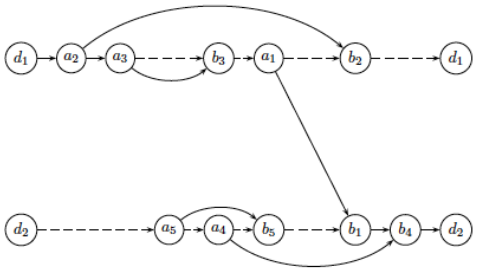
Complexity: As elaborated, the problem is modeled as a kind of a vehicle scheduling problem. Therefore, it is easy to show that the MMCRP is NP-hard if the number of depots is at least two. We prove the complexity by reduction from the *vehicle scheduling problem with 2 depots* which was proven to be NP-hard by Bertossi et al. [24]. Notice that, since the vehicle scheduling problem does not consider ride-sharing, the MMCRP with at least 2 depots is NP-hard even without ride-sharing.

3.4.1 Arc formulation

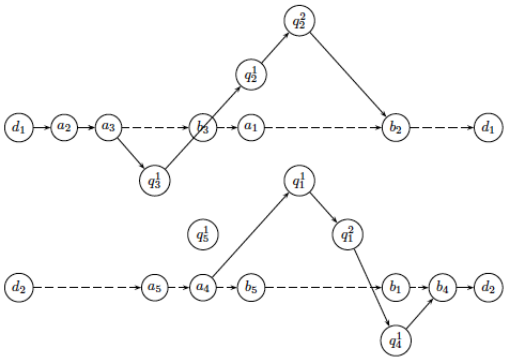
We first introduce a direct formulation of the MMCRP based on the auxiliary graph $G = (V, E)$ presented in Section 3.3. For every node $v \in V$ we have the set of outgoing arcs E_v^+ and ingoing arcs E_v^-. Let V' be the set of intermediate nodes $V' = V \setminus \{(d, \sigma), (d, \tau)\}_{d \in D}$. The set E^q denotes all arcs e that cover task q, including co-riding visits. Q denotes the set of all tasks. Finally let γ_e denote the savings of arc e. The binary decision variable x_e takes on value 1 if arc e is selected in the solution and 0 otherwise.



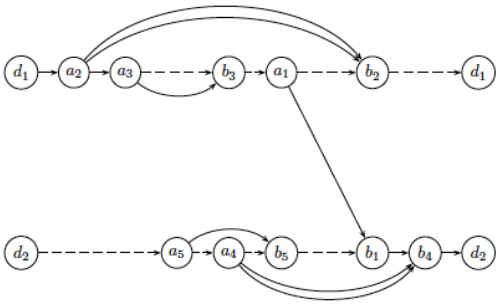
(a) First step of the graph construction: graph including information on tasks.



(b) Second step of the graph construction: modeling the trips as arcs.



(c) Graph with two ride-sharing trips.



(d) Extended graph with trips as arcs, where two ride-sharing trips are added to the graph in (b).

Figure 3.3: Illustration of the auxiliary graph in a time-space network.

$$\max \quad \sum_{e \in E} \gamma_e x_e \tag{3.3}$$

$$\text{s.t} \quad \sum_{e \in E_v^-} x_e = \sum_{e \in E_v^+} x_e \quad \forall v \in V' \tag{3.4}$$

$$\sum_{e \in E_{(d,\sigma)}^+} x_e = W_d \quad \forall d \in D \tag{3.5}$$

$$\sum_{e \in E_{(d,\tau)}^-} x_e = \overline{W}_d \quad \forall d \in D \tag{3.6}$$

$$\sum_{e \in E^q} x_e \leq 1 \quad \forall q \in Q \tag{3.7}$$

$$x_e \in \{0,1\} \quad \forall e \in E \tag{3.8}$$

The objective (3.3) maximizes total savings over all arcs. Constraints (3.4) ensure flow conservation at intermediate nodes $v \in V'$. Constraints (3.5) and (3.6) ensure that there is a correct number of vehicles $W_d, \overline{W}_d$ at start and end of the time horizon for each depot $d \in D$. Constraints (3.7) make sure that each task $q \in Q$ is covered at most once. If a given task is not covered, the assigned user will reach the task using the cheapest other MOT.

The model has $O(E)$ variables, and $O(V + D)$ constraints. Hence it is polynomial in the size of the graph. However, the graph $G = (V, E)$ may be large (exponential in the original input size) due to the number of possible co-rides.

Despite the compact arc formulation, but due to the size of the graph, we will see in the computational experiments that only relatively small problems can be solved using this model. We will therefore introduce a stronger but larger formulation based on a path formulation.

3.4.2 Path formulation

In order to introduce a path formulation of the MMCRP, we assume that all possible routes ρ of all vehicles are enumerated in the set $\mathcal{R}$. Each route ρ must start in node (d,σ) and finish in a node (d,τ), traversing arcs E in the auxiliary graph $G = (V, E)$. The start and end depots d may be different.

Let γ_ρ be the savings of route ρ calculated as the saving by using a car compared to the cheapest other MOT for all arcs on the respective route. Furthermore, let the binary matrix $F_{\rho q}$ be 1 if route ρ will service task q. Finally, let $G_{\rho d} = 1$ if route ρ starts in depot d, and $H_{\rho d} = 1$ if route ρ ends in depot d and 0 otherwise. The values W_d and $\overline{W}_d$ state the number of vehicles that are available at depot $d \in D$ at the beginning and end of the planning horizon. The binary decision variable x_ρ takes on value 1 if route ρ is chosen, and 0 otherwise.

We can now formulate the MMCRP as follows:

$$\max \quad \sum_{\rho \in \mathcal{R}} \gamma_\rho x_\rho \tag{3.9}$$

$$\text{s.t} \quad \sum_{\rho \in \mathcal{R}} F_{\rho q} x_\rho \leq 1 \qquad q \in Q \tag{3.10}$$

$$\sum_{\rho \in \mathcal{R}} G_{\rho d} x_\rho = W_d \quad d \in D \tag{3.11}$$

$$\sum_{\rho \in \mathcal{R}} H_{\rho d} x_\rho = \overline{W}_d \quad d \in D \tag{3.12}$$

$$x_\rho \in \{0,1\} \qquad \rho \in \mathcal{R} \tag{3.13}$$

The objective function (3.9) maximizes the sum of the savings of the selected routes. Constraints (3.10) make sure that each task $q \in Q$ is covered at most once. Constraints (3.11) make sure that exactly W_d vehicles leave depot d at the start of the planning horizon, and constraints (3.12) make sure that exactly $\overline{W}_d$ vehicles return to depot d at the end of the planning horizon. Since not all vehicles have to be used, we add the necessary dummy routes to the set $\mathcal{R}$. Finally, constraints (3.13) define the decision variables x_ρ to be binary.

3.4.3 Delayed column generation

Since the number of routes $\mathcal{R}$ in model (3.9)-(3.13) may be very large we solve its LP-relaxation through column generation.

The restricted master problem considers a subset of routes $\mathcal{R}' \subseteq \mathcal{R}$ of all possible routes. In every iteration a pricing problem is solved to find a new route with positive reduced savings. The process is repeated until no more routes with positive reduced savings can be found. When the process terminates, we have solved the LP-relaxation of (3.9)-(3.13) and hence have an upper bound on the solution to the MMCRP.

The pricing problem is searching for a route through the auxiliary graph $G = (V, E)$ maximizing the reduced savings. The problem becomes a kind of (time constrained) shortest-path problem in G where we aim at finding the paths with the largest savings. Since G is a time-space network, the time constraints are implicitly handled by graph construction. Moreover, we note that G is a DAG, and hence no cycles can occur.

Let y_q be the dual variable corresponding to task covering constraint (3.10), u_d be the dual variable corresponding to depot start-inventory constraint (3.11), and $\overline{u}_d$ be the dual variable corresponding to depot end-inventory constraint (3.12). The reduced savings of a route ρ starting at depot d and ending at depot d can be calculated as follows:

$$\sum_{q \in \rho} (\gamma_q - y_q) - u_d - \overline{u}_d \tag{3.14}$$

The function sums all savings γ_q of tasks q covered by route ρ subtracted by the dual variables y_q. Finally the dual variables $u_d, \overline{u}_d$ corresponding to the depot inventory constraints are subtracted.

3.4.4 Pricing problem

The pricing problem generates new promising routes by finding a path with the maximum reduced savings in the time-space network $G = (V, E)$. We find the path using a label setting algorithm.

The pricing problem is solved for each combination of start depot $d \in D$ and end depot $d \in D$. Promising routes with positive reduced savings are added to the master problem until no more routes with positive reduced savings can be found.

Dynamic programming

We solve the pricing problem for every pair of start and end depot, using auxiliary graph $G = (V, E)$. We use a label setting algorithm adapted to a DAG. For every node $v \in V$, we have an associated value f_v denoting the path with the so far largest savings to v. Initially $f_v = -\infty$ for all nodes $v \in V$ except the source node, and gradually the value of f_v is increased as paths with higher savings are encountered.

A dynamic program is solved for each pair of depots. However, for each start depot, the dynamic programming algorithm will actually solve the problem for all destination depots. So, we only need to call the dynamic programming algorithm $|D|$ times, resulting in the overall time complexity $O(|D| \times |E|)$ for solving all pricing problems.

Since we have one value f_v for every node, the space complexity of the dynamic programming algorithm is $O(|V|)$. This is clearly overshadowed by the size of the graph.

Notice that when the dynamic programming algorithm terminates we may have several distinct solutions ending at depot d, for different arrival times h. The algorithm may easily be modified to return all these distinct solutions.

Stopping criterion and columns added

In the basic pricing algorithm we run the pricing for each combination of depots such that the column with the most positive reduced savings is added. We denote this strategy as `best`. Alternatively, we also evaluate the following strategies: `first`, `firstdep`, and `multiple`. In `first` we stop as soon as a column with positive reduced savings has been found and add this column to the master problem. Again, in each iteration only one column is added. Next, we extend `first` for every depot combination, and we iterate until the first column with positive reduced savings is found for each combination and terminate thereafter. This means that, when considering two depots and combining each of them, we have at most 4 columns added in each iteration. This is denoted as `firstdep`. Lastly, in `multiple` we include all columns with positive reduced savings. Note that we also tried to restrict the number of columns added. However, we did not see a remarkable difference to the non-restricted case.

Heuristic pricing algorithm

Although the pricing problem is solvable in polynomial time in the size of the graph $G = (V, E)$, the number of arcs E may be very large, and we will see in the computational experiments that the pricing problem takes up most of the solution time. We therefore introduce a number of heuristic pricing algorithms, namely `statespace`, `heurprun` and

`heurarcs`. When employing one of the heuristic pricing strategies, we search for columns with positive reduced savings and afterwards finish with one of the exact pricing schemes.

`statespace`: In this method, we reduce the auxiliary graph by merging nodes with similar time in the time-space graph. The time-horizon is discretized in intervals of 10 minutes. If two nodes, corresponding to the same depot, end up in the same time interval, they are merged. After the merging, there may be multiple arcs between each pair of nodes, so we select the arc with highest savings, and ignore the rest.

`heurprun`: In this method, we use an aggressive reduction of the graph, by only keeping the savings and not the time. Hence, for every set of ride arcs $E_p = \{((a_p^\pi, g_p^\pi), (b_p^\pi, h_p^\pi))\}_{\pi \in R}$ we merge start and end times, g_p^π and h_p^π, into one common artificial time g'_p and h'_p for each user's start and end depot a_p and b_p such that, to start with, we only keep track of the ride with largest savings $\pi \in R$ for each user p.

`heurarcs`: In the original algorithm we construct an auxiliary graph $G = (V, E)$ in which we may have arcs $e \in E$ with both positive and negative savings γ_e. Arcs having a negative saving will only be used if they can be combined with arcs having a positive saving. The heuristic pricing algorithm removes all arcs e with negative savings $\gamma_e < 0$ before running the label setting algorithm. This reduces the size of the auxiliary graph.

3.5 Computational study

The algorithms are implemented in C/C++ and for the solution of the master problem CPLEX 12.6.2 together with Concert Technology 2.9 is used. Tests are carried out using one core of an Intel Xeon 2643 machine with 3.3 GHz and 16 GB RAM running Linux CentOS 6.5. The algorithms are tested on a number of generated instances of increasing size and complexity. Various pricing schemes are compared, and the efficiency of all parts of the code is evaluated. Most of the reported computation times include the generation of the auxiliary graph. The exception is the comparison of the arc formulation to the column generation approach, where only the time to solve the models is stated.

To start the column generation we provide an initial set of dummy columns by inserting route variables x_ρ such that the master problem is feasible without considering any valid route construction. These dummy variables are leaving and entering a depot (constraints (3.11)-(3.12)), but do not cover any tasks. We first solve the linear relaxation of the master problem and thereafter we solve the restricted master problem to integer optimality, using only routes generated in $\mathcal{R}'$, containing a subset of all routes R with positive reduced savings. In this way we get an upper bound from the column generation based approach, and a lower bound from solving the IP model. Although we cannot guarantee an optimal solution to the original MMCRP in this way, the results will show that in most cases the gap is very small, and the solution quality is more than sufficient for practical applications.

In the following, we first introduce the test instances in Section 3.5.1. In Section 3.5.2 we compare the pricing schemes introduced in Section 3.4.4 and afterwards conduct algorithmic tests in Section 3.5.3. Finally, we discuss socio-economic aspects in Section 3.5.4. For a better understanding of the problem we provide a sample solution in the Appendix A.2.

We show in our results that the column generation approach is an efficient choice to solve the MMCRP. We can solve the biggest instances with 300 users and 40 ve-

hicles within less than one hour on average. Pricing strategy `multiple` turns out to be the most efficient of the exact methods. The heuristic approaches (`statespace`, `heurprun`, `heurarcs`) do not come with a significant improvement in solution time or quality. Finally, we compare our results to those of the direct formulation presented in Section 3.4.1.

3.5.1 Test instances

We generate realistic benchmark instances based on available demographic, spatial and economic data of the city of Vienna, Austria. Five different MOTs are considered: car, walk, bike, public transport, and taxi. Walk, bike, public transport, and taxi are assumed to have an unrestricted capacity $m_k = \infty$, while there is a limited number $m_{car} < \infty$ of shared cars. For each mode of transport $k \in K$ we define a set of properties, described in the following. Information of the car is based on available data (PKW-Mittel Diesel in Beermann et al. [19]). Distance d_{ij}^k, time w_{ij}^k and cost c_{ij}^k are calculated between all nodes i and j for all modes of transport $k \in K$. Average travel speed per transport mode are as follows (in km/h): car = 30, walk = 5, bike = 16, public transport = 20, taxi = 30.

Emissions ϵ_{ij}^k are translated into costs and, together with distance cost c_{ij}^k and cost of time $w_{ij}'^k$, included into the overall cost calculations. Costs per emitted ton of CO_2 is 5€ and average gross salary in Austria including additional costs for the employer is 19.42€/hour. Variable cost per distance c_{ij}^k of the car is taken from the available car information in Beermann et al. [19] and is 0.188€/km. For taxi we take on a value of 1.2€/km. As we only consider distance cost that are variable and no fixed charges, we assume for all other MOTs costs c_{ij}^k of 0. Additional time ξ^k is added to denote extra time needed for a certain MOT k, such as additional 10 minutes for cars to account for walking distances from/to the parking lot. This we specify as follows (in seconds): ξ^{car} = 600, $\xi^{walk} = 0$, $\xi^{bike} = 120$, $\xi^{public} = 300$, $\xi^{taxi} = 300$. Distances are based on aerial distances and multiplied by a constant sloping factor ζ^k for each mode k in order to account for shortcuts/detours usually associated with certain modes of transport. These we define as $\zeta^{car} = 1.3$, $\zeta^{walk} = 1.1$, $\zeta^{bike} = 1.3$, $\zeta^{public} = 1.5$, and $\zeta^{taxi} = 1.3$.

Each generated instance represents a distinct company consisting of one or more depots $d \in D$ and users, i.e., employees, $p \in P$. The locations of the depots are based on statistical data of office locations in Vienna. The set of possible locations is based on geometric centers of all 250 registration districts of Vienna.

Companies are defined by a fixed number of users u and depots $|D|$. The number of customer visits, i.e., meetings, and their time and location, are then randomly generated based on historic statistical data.

To each user p we associate a subset of MOTs $K^p \subseteq K$. We assume penalties for constraint violations such as choosing a mode of transport that is not in the user's choice or for too late arrival. The penalty cost per violation is determined to be 10,000 and directly included into the cost function.

For our calculations, we depict one day only. Each user p has an assigned set of tasks Q^p distributed over the day. For the smallest instances on average 95 nodes (comprising meetings/tasks q and start and end depots a, b) and 33 tasks are generated. For the largest instances we have on average 463 tasks. Further information is given in

Section 5.4 in Table 3.7. This leads to about 4-5 assigned nodes per user (including their start and end depots of the trips). The ordered list of tasks Q^p for a working day per user p is generated with the following attributes for each task q: latest arrival, earliest departure, service duration, all given in minutes. We already account for ride-sharing in the instance generation where we enforce the proximity of various tasks of different users. First, a predefined sequence of tasks is generated per user p which is then partitioned into separate sets of tasks with newly assigned artificial user $p' \in P$ if a user returns to the depot more than once during a day. If a user p has more than one simple trip, buffer time at the depot is set to 60 minutes in order to account for, e.g., changing of cars or additional time needed if the previous route was not covered by car. We assume that the maximum distance between two nodes is one hour.

Instances are named as `E_`u`_`I, where u is the number of users, and the instance number I is between 0 and 9. For example, the first instance in the set of instances with 20 users ($u = 20$) is denoted `E_20_0`. For each combination of u and m we solve a set of 10 instances (`E_u_0` to `E_u_9`) and report the average values.

Instance sets and the source code of the instance generators are made publicly available at `https://github.com/dts-ait/seamless`.

3.5.2 Comparison of the different pricing schemes

In this part, we compare different pricing schemes and study heuristic pricing algorithms. We compare four different variants of how and when to add columns to the master problem (described in Section 3.4.4) and three heuristic approaches, as described in Section 3.4.4. We provide insights into different parts of the algorithm and finally choose the variants having the best trade-off between solution time and solution quality. We compare results based on an increasing number of users $u = 20, 50, 100, 150, 200, 250, 300$ and vehicles $m = 2, 4, 10, 20, 40$. In our experiments the number of depots is two except for Table 3.6 where we study instances with more depots. The vehicles are equally split over all depots.

To start with, we study the solution time and solution quality of the different exact pricing schemes described in Section 3.4.4. Notice that all pricing schemes return the same LP-bound, but the IP-solution may be different because a different set of columns may be generated.

In Table 3.1 we report computational times in seconds and the average gap in percentages between the integer and LP solution for the respective set of instances. The gap between the two solutions is calculated as: (Savings LP − Savings IP)/Savings IP. We split the table into combinations of pricing scheme (`best`, `first`, `firstdep`, `multiple`), number of vehicles ($m = 2, 4, 10, 20, 40$) and users ($u = 150, 200, 250$). Note that for `multiple` we add a row for $u = 300$ as this is the only exact scheme that was able to solve all instances within the stated time limit of two hours. As it may be seen, we are able to find near optimal solutions with a gap close to 0. For the case with $m = 2$, which means one vehicle for each depot, we can close the gap for all instances given in the table. The gap increases slightly when more vehicles are added, however only up to a certain point. For instances with more cars ($m = 10, 20, 40$) we cannot see a significant difference in the gap anymore. All approaches return solutions of approximately equally good quality. However, strategy `multiple` gives slightly better gaps than the other schemes.

In schemes `best`, `first`, and `firstdep` only a very restricted subset of columns is added to the problem in each iteration. However, they might not be the most beneficial for the integer solution. With `multiple` we add all found columns with positive reduced savings which helps us to identify an integer solution. Given this situation and also for practical reasons, we decided to refrain from implementing a full fledged branch-and-price algorithm. In terms of computational time, for instances with a small number of cars ($m = 2, 4$) and up to 150 users ($u = 150$) it does not make a difference which scheme is used. By increasing the number of users and keeping a small fleet we can gradually see a difference. Pricing scheme `best` performs worst and pricing scheme `multiple` is, by far, the most efficient in terms of computation time. For the largest comparable instance which can be solved by all schemes ($m = 40$ and $u = 250$), computation times differ by a factor of 7: the average run time of pricing scheme `multiple` amounts to 843 seconds and to 6,145 seconds with pricing scheme `best`. The biggest instances ($u = 300, m = 40$) can be solved within less than one hour on average. We can observe in our computational results that the time spent on solving the master problem is usually very small, below three minutes on average, with the exception of instance class $u = 100$ and $m = 10$, where we obtain an average value of 1,296 seconds, using pricing scheme `first`. Most of the computation time is spent on solving the pricing problem. Observing this, we try to decrease computation times by studying three different heuristics to accelerate pricing, namely `heurarcs`, `heurprun`, `statespace` (see Section 3.4.4).

Table 3.1: Average computation time in seconds and average gap in percentages (%) between the LP solution and the integer solution for the exact pricing schemes for an increasing number of users (u) and vehicles (m). The gap is calculated as: (Savings LP − Savings IP)/Savings IP.

		$m = 2$		$m = 4$		$m = 10$		$m = 20$		$m = 40$	
	u	time	%	time	%	time	%	time	%	time	%
`best`	150	4.7	0.00	17.3	0.46	97.1	1.08	421.9	0.25	845.6	0.14
	200	9.1	0.00	37.1	0.21	328.2	0.45	2797.6	0.18	2805.8	0.23
	250	18.9	0.00	83.4	0.07	1799.7	1.05	2738.6	1.07	6145.4	0.18
`first`	150	4.1	0.00	12.1	0.46	58.8	0.83	254.4	0.23	582.6	0.08
	200	7.6	0.00	28.9	0.21	355.3	0.43	1396.7	0.22	2058.1	0.15
	250	16.5	0.00	61.7	0.07	1321.7	0.79	3512.2	1.11	4506.4	0.15
`firstdep`	150	4.4	0.00	14.6	0.47	86.7	0.98	242.2	0.26	518.5	0.11
	200	7.7	0.00	28.4	0.21	277.8	0.49	1307.4	0.22	1690.5	0.18
	250	16.9	0.00	64.3	0.07	625.6	0.98	1677.9	1.18	3303.6	0.16
`multiple`	150	4.0	0.00	9.3	0.46	28.3	0.63	72.2	0.24	114.6	0.04
	200	8.3	0.00	20.4	0.21	108.3	0.44	270.7	0.23	362.6	0.10
	250	17.7	0.00	42.3	0.07	222.1	0.57	493.7	0.75	842.7	0.12
	300	50.1	0.00	156.4	0.70	631.5	0.59	1734.4	0.11	2994.6	0.16

All heuristic approaches use pricing scheme `multiple` as it turned out to be the most efficient of the introduced exact pricing algorithms. The heuristic pricing schemes

are employed as follows: We use the heuristic as long as columns with positive reduced savings can be found. Thereafter, we continue with the chosen exact pricing procedure.

Table 3.2 gives the average gap between the upper bound and solving the original IP on the same set of columns, and the average computation times in seconds for the heuristic pricing schemes `heurarcs`, `heurprun`, `statespace`, and scheme `multiple` for the larger instance classes defined by $u = 150, 200, 250, 300$ and $m = 2, 4, 10, 20, 40$. We see for all schemes a gap below 1%, and similar computational times. Thus, none of the presented schemes significantly stands out in terms of running times or solution quality.

Table 3.2: Average computation time in seconds and average gap in percentages (%) between the LP solution and the integer solution for the heuristic pricing schemes and scheme `multiple` for $u = 150, 200, 250, 300$ and an increasing number of vehicles m. The gap is calculated as (Savings LP − Savings IP)/Savings IP and reported for increasing number of vehicles (m).

	$m = 2$		$m = 4$		$m = 10$		$m = 20$		$m = 40$	
	time	%	time	%	time	%	time	%	time	%
`multiple`	20.0	**0.00**	57.1	0.59	247.5	**0.45**	642.8	**0.21**	1078.6	0.10
`heurarcs`	**18.4**	**0.00**	48.5	0.21	**218.7**	0.53	**571.1**	0.44	1075.2	0.10
`heurprun`	22.2	**0.00**	52.3	0.20	229.4	0.57	616.6	0.50	**1036.7**	**0.09**
`statespace`	19.1	**0.00**	**47.6**	0.23	224.9	0.57	571.9	0.43	1060.7	0.11

In Table 3.3, we present the total number of columns generated when running the respective pricing algorithm over all instances with $m = 2, 4, 10, 20, 40$. We observe that, as expected, by only adding one column in each iteration (`best`, `first`) and one for each depot combination (`firstdep`) we generate fewer columns than with scheme `multiple`, which adds all columns with positive reduced savings.

Table 3.3: Total number of columns generated by the different pricing schemes over all instance classes with a given number of vehicles (m).

	$m = 2$	$m = 4$	$m = 10$	$m = 20$	$m = 40$
`best`	28	149	855	2,771	4,993
`first`	30	163	13,903	3,883	6,606
`firstdep`	29	154	857	2,585	4,710
`multiple`	263	645	2,751	6,797	12,865

The solutions gained when solving the arc formulation and the integer solutions obtained by the column generation algorithm are compared in Table 3.4. We run the arc formulation on a set of small instances ($u = 20, 50, 100$ and $m = 4$) already showing the benefits of the decomposition algorithm. In the first row (computation time (s)) the average computation times to solve the arc formulation (AF), and the column generation algorithm (CG) are given. The stated numbers encompass only the computation times in seconds to solve the respective formulation, thus the enumeration of the trips is excluded. We observe that for the smallest instances the arc formulation is faster. However, for the bigger instances ($u = 50, 100$), we need a multiple of the time of the column generation algorithm. The gap (gap (%)) between the solution of the arc formulation and the respective integer solution is very low, less than 0.11% for all instances on average.

With the column generation approach we are able to solve 9 out of 10 instances of the respective instance group to optimality (# opt.)

As we assumed and also show in Table 3.7, the number of arcs generated, and thus the underlying graph, becomes very large and therefore this formulation cannot be used efficiently in a direct formulation. Note that we also tried to run the arc formulation by increasing the relative MIP gap (e.g., to 1%). This, however, did not lead to any improvements as solving the root relaxation takes most of the computational time.

Table 3.4: Average time in seconds for solving the arc formulation and the column generation based algorithm, average gap in percentages between the solution of the arc formulation (AF) and integer solution of the column generation based algorithm (CG), and number of instances solved to optimality with the column generation based algorithm for instances with an increasing number of users $u = 20, 50, 100$ and number of vehicles $m = 4$.

$u=$	20		50		100	
	AF	CG	AF	CG	AF	CG
computation time (s)	0.1	0.6	23.1	0.8	1,615.9	1.2
gap (%)	0.00	0.06	0.00	0.11	0.00	0.03
#opt.	10/10	9/10	10/10	9/10	10/10	9/10

3.5.3 Algorithmic tests

In the following we use configuration `statespace` to study how the column generation evolves, the impact of early termination of the column generation and different numbers of depots.

In terms of convergence of the algorithm, we observe the common picture of a steep increase in the objective value during the first iterations and then a long tail until optimality of the LP relaxation has been proven. This means that we are able to find good solutions close to the optimal objective function value in a relatively few iterations. However, the column generation then needs quite a number of iterations in order to find the optimal value. This effect can be exploited for practical applications: the column generation process can be terminated early without loosing much in terms of solution quality. Figures 4(a), (b), (c) and (d) in the Appendix A.2 plot the convergence of the column generation algorithm. The number of iterations is shown on the x-axis and the objective function value on the y-axis. We report results for $u = 150$ users (a), $u = 200$ user (b), $u = 250$ users (c), and $u = 300$ users (d). Each curve represents the outcome of one instance.

In Table 3.5 the impact of early termination after about one third of the iterations on the quality of the obtained objective value is shown. As the number of iterations needed to find the optimal solution does not vary much between instances of different size, we assume a common termination criterion for all. Observing that usually about 140-160 iterations are needed to terminate the algorithm, we study early termination after 50 iterations and solve the integer problem on the columns generated so far. Table 3.5 gives the gap in percentages between the integer and LP solution for each instance with $u = 300$. The row "time (s)" shows the average run time in seconds. We observe that we are still able to find good solutions after only one third of the iterations. The computed

gap between the original upper bound and solving the IP on the restricted set of columns, using early termination, is at most 6.6% and 2.5% on average, which is good enough for practical applications. Since the algorithm is stopped after about one third of the previously necessary iterations, run times are reduced accordingly.

Table 3.5: Impact of early termination of the column generation algorithm after 50 iterations for $u = 300$ and $m = 40$. Comparison between original upper bound and obtained integer solution after early termination. Gap in percentages and time in seconds are given for the case of early termination (terminate) and the original values obtained from `statespace`.

		E_300_0	E_300_1	E_300_2	E_300_3	E_300_4	E_300_5	E_300_6	E_300_7	E_300_8	E_300_9	average
gap (%)	terminate	6.45	2.56	2.29	0.19	2.70	2.82	0.16	1.80	4.52	2.55	2.60
	original	0.00	0.76	0,78	0.00	0.72	0.00	0.00	0.61	0.00	0.06	0.16
time (s)	terminate	637.9	506.3	101.1	958.2	1,193.0	119.7	905.7	69.8	1,325.3	1,101.3	691.8
	original	5,533.2	3,198.1	424.8	1,043.4	5,430.3	402.5	1,051.5	328.2	7,340.5	5,194.0	2,994.6

In Table 3.6 we show the impact of increasing the number of depots on the run time of the algorithm. In all previous experiments, two depots were used. We now use 1, 3, and 4 depots. Within the project, the case of 1 depot is chosen as currently a company usually operates one main office (or at maximum two) with shared cars. However, as we are investigating the future sharing economy and different settings of companies we also analyse if our algorithm is capable of dealing with more depots. The number of vehicles is shown as the number of vehicles per depot to allow for a fair comparison. This means that, to obtain the actual total number of vehicles, this number must be multiplied by the number of depots. As expected, computation times increase with rising number of depots, however only to a certain factor and not exponentially. As the pricing algorithm is solved for each pair of depots we are increasing the number of subproblems the pricing algorithm is able to handle. However, as only a few trips start and end at different depots, we can provide reasonable solution times even for the case of four depots. In Table 3.6 the respective values are provided.

Table 3.6: Average computation times in seconds for 1, 2, 3 and 4 depots for users $u = 20, 50, 100, 150, 200, 250, 300$ and vehicle per depot (m').

	$m' = 1$	$m' = 2$	$m' = 5$	$m' = 10$	$m' = 20$
1 depot	8.4	15.0	62.4	134.3	372.3
2 depots	12.3	33.7	143.4	370.0	619.2
3 depots	15.6	54.9	208.5	586.7	652.2
4 depots	35.9	143.2	554.2	1105.2	1601.6

3.5.4 Socio-economic tests

In this section, we summarize the results of our socio-economic tests. All results are obtained using the version of our algorithm with the smallest LP-IP gap, which is `multiple`. We study savings by ride-sharing, savings by car-sharing and give insights into the instances and strategic decisions regarding the optimal size of the car pool.

Table 3.7 gives the average number of all trips for each instance class and the average number of simple trips. Simple trips are the arcs without any ride-sharing activities going from start node a to end node b. As we can see, the number is always slightly higher than the number of users, indicating that a set of users return to the depot during the day and start another sequence of nodes in the observed planning horizon. Column "tasks incl. start/end" gives the average number of nodes, i.e., tasks and start/end depots of a user's trip, of the instance set. Each user covers about 4-5 nodes, each trip includes on average 3.3 nodes. The column "tasks" gives the average number of tasks for each instance set. This number ranges from 33 ($u = 20$) to 463 ($u = 250$), which gives approximately 1-2 tasks per trip. In the last four columns we give the percentage of tasks and start/end depots covered by a car. Naturally, with a low number of cars ($m = 4$) and a high number of users ($u = 250$), only a small subset of tasks will be covered by car. The coverage ranges from 4% to 45%.

Table 3.7: Number of simple trips, number of all trips, and tasks in the auxiliary graph for an increasing number of users u. The columns on the right give the percentage of nodes including tasks, start and end depots, covered by car for $m = 4, 8, 20, 40$.

u	simple trips	all trips	tasks incl. start/end	tasks	$m =$ 4	8	20	40
20	31	240	95	33	30%	38%	41%	41%
50	76	7,403	242	90	13%	25%	42%	43%
100	147	71,726	476	183	9%	15%	35%	45%
150	218	551,936	714	279	6%	11%	25%	43%
200	287	1,497,545	951	381	5%	9%	20%	37%
250	358	2,317,145	1,179	463	4%	7%	16%	30%

In Table 3.8, the results obtained from solving the MMCRP are compared to only car-sharing (ratio (1)) and user-dependent car assignment (ratio (2)). The value is given as the savings ratio of MMCRP : car-/ride-sharing.

User dependent car-assignment means that if a user has an assigned car, the selected user will have the car for the whole day and use it for all trips. Moreover, no other user is allowed to use this car and ride-sharing is not possible. If only car-sharing is employed users may hand over the cars during the planning horizon so that a car will have different drivers assigned, however, no ride-sharing is allowed. All tests are run for an increasing number of vehicles ($m = 2, 4, 10, 20$) and users ($u = 20, 50$). Instance specific results are reported and summed up in the row "average".

By allowing car- and ride-sharing and thus having a more flexible usage of the car pool rather than a restricted usage during a day, we can have up to 1.7 times higher savings in the planning horizon. This is already shown for small-sized instances. As expected, the more flexible the usage of the cars, the more savings are achieved. Please note that instances E_20_4, E_20_7, E_20_8, and E_50_1 are not in the table, meaning that these are not included in the average calculations as they would give a somewhat misleading outcome. This is due to the fact that we assumed penalties for constraint violations such as choosing a mode of transport that is not in the user's choice or for too late arrival. The penalty cost per violation is determined to be 10,000 and directly included into the cost function. Therefore we obtain for some instances very high savings which is mainly due to these penalties included in the objective function. Note that we

also excluded values for $m = 20$ and $u = 20$ as this setting means that more cars than users are provided.

Table 3.8: Increase in savings when comparing car-sharing (without ride-sharing), user-dependent car-sharing (i.e., car-sharing without ride-sharing and a user has a car for the whole day) and MMCRP. Ratio (1) reports the factor of improvement in savings in comparison to car-sharing, ratio (2) reports the enhancement when comparing to the user-dependent car-sharing.

	$m = 2$		$m = 4$		$m = 10$		$m = 20$	
	ratio (1)	ratio (2)	ratio (1)	ratio (2)	ratio (1)	ratio (2)	ratio (1)	ratio (2)
$u = 20$	1.2	1.4	1.2	1.4	1.3	1.3	-	-
$u = 50$	1.3	1.6	1.5	1.6	1.6	1.7	1.7	1.7

When analyzing fleet size, we see initially big savings when adding more vehicles yet the impact diminishes quite fast. For instances with 20 users fewer than 5 vehicles suffice, for 50 users fewer than 10 are certainly enough and when we consider 100 users the breaking point is somewhere between 20 and 30. For larger instances ($u = 150, 200, 250, 300$), the ideal number of vehicles would be between 20 and 50. Figure 5 in the Appendix A.2 provides illustrative insights into the optimal fleet size for an increasing number of users employed ($u = 20, 50, 100, 150, 200, 250, 300$). The x-axis represents number of vehicles, the y-axis the objective function value and each line represents a distinct instance of our experiments, whereas the thicker black line in each subfigure shows the average.

Finally, we analyse the average number of arcs and ride-shares in our results. Table 3.9 summarizes the average number of trips per vehicle route and average ride-sharing activities per arc (ride-sharing per ride). This is shown for the cases of 20 and 50 users and increasing number of vehicles ($m = 2, 4, 10, 20$ respectively). The average number of trips on a vehicle route gives us an idea of the amount of car-sharing activities. With an increasing number of cars the number of trips on a vehicle route is decreasing. The very small numbers, for example 0.9 for $u = 20$ and $m = 10$, are mainly due to unused arcs, meaning that not all of the available cars are used. Moreover, we have on average 1.5-1.7 ride-sharing activities per arc, which is considered to be very high and supports our goal of having a good utilization of the pool of cars. Note that we again exclude values for $m = 20$ and $u = 20$ as this setting would mean that more cars than users are provided.

Table 3.9: Average number of trips per car, and average number of ride-shares per trip.

	$u = 20$		$u = 50$	
m	trips per car	ride-sharings per trip	trips per car	ride-sharings per trip
2	2.0	1.5	2.1	1.6
4	1.6	1.5	1.9	1.5
10	0.9	1.2	1.7	1.7
20	-	-	1.1	1.7

3.6 Conclusion

Inspired by the concept of sharing economy and future mobility systems, we introduced the multimodal car- and ride-sharing problem (MMCRP) that assigns different modes of transport to ride requests. We aimed at deploying a pool of shared cars as efficiently as possible, join ride requests by offering ride-sharing and by assigning different modes of transport to the remaining requests.

We introduced the novel MMCRP and showed that the problem is NP-hard if the number of depots is at least two. The problem remains NP-hard even if ride-sharing is not allowed. In order to circumvent the complexity of modeling ride-shares and additionally assigning further modes of transport, we constructed an auxiliary graph in which all possible ride-sharing rides are enumerated. Ride requests not covered by a car or ride-share are appointed to take the cheapest other MOT. This made it possible to formulate a compact model for the problem as a kind of vehicle scheduling problem. We extend the vehicle sharing problem by allowing for different start and end depots. Moreover, due to the modeling of ride-sharing into the graph, we may have multiple possibilities to cover a trip and have to make sure that a user is not riding in parallel, i.e., that a user is not driving in multiple cars at the same time. Note that the auxiliary graph can be exponential in the original input size. Due to the size of the auxiliary graph, the compact model is also quite complex to solve, hence we proposed a path-based formulation. To solve the path-based formulation we introduced an efficient two-stage decomposition algorithm and relied on well-studied approaches to solve the real-world problem. In the first stage of the decomposition approach, trips were enumerated and afterwards, in the second stage, solved through a column generation approach. The master problem keeps track of the requests and depot balance constraints, while the pricing problem generates improving paths. We showed that the pricing problem can be solved through dynamic programming in polynomial time, measured in the size of the auxiliary graph. The computational results confirmed that large instances can be solved in reasonable time, making it possible to use the algorithm for daily planning of multimodal car- and ride-sharing problems even in a large-scale setup. The two-level decomposition makes it easy to implement additional constraints on co-riding to make it more attractive: This could be limits on detour or driving time, or co-rider preferences. Also the framework can easily be generalized to handle more than one co-rider, which should also be done in future work.

The introduced model targets corporate mobility services. However, it can easily be applied to any specific network with a predefined set of users in a closed community and is therefore of high importance in current and future concepts of shared mobility systems. Moreover, the MMCRP can be extended to electric cars, where the algorithm must ensure that sufficient time is available at the depot for recharging the cars. Furthermore, the model can be extended to consider several shared modes of transport. This could be electric as well as conventional cars and bikes that are pooled to satisfy the needs of transportation for one or more companies. Bikes might need to be embedded in a rebalancing system, ensuring that the right number of bikes is at the right locations at the beginning and end of the day. Bikes can thus be implemented in a unified model with other shared MOTs considering specific limitations as they cannot be available for ride-sharing. However, as we are considering urban mobility, we can assume that

there are shared bikes provided from different independent providers, and therefore this mode of transport is always available for the users and the company does not have to plan the rebalancing on their own. Furthermore, the current model is restricted to a predetermined fixed sequence of tasks. This was considered as given from our practical partners. However, this restricts the model and prevents possible further savings that might benefit from this flexibility. Moreover, the model might profit from allowing changes of drivers on a trip. We note that this restriction was introduced based on information from our industry partners. They reported that there was very limited acceptance for handing over cars during a trip. However, in our future work we plan to address this aspect as well as more flexibility in the timing of user tasks. Lastly, another interesting aspect could be the consideration of uncertain and varying travel times. While time-dependent travel times would not require major changes to our approach, since we already work on a time expanded network, the consideration of uncertainty would require major changes in the design of the solution approach. Furthermore, an unexpected delay of the driver at any point would affect any later co-rider on the route, requiring changes in the schedule during the day, which should also be addressed in future work.

Acknowledgements

This work was supported by the Climate and Energy Funds (KliEn) [grant number 853767]; and the Austrian Science Fund (FWF): P 31366.

Chapter 4

The bi-objective multimodal car-sharing problem

Accepted for publication at: *OR Spectrum*
April 8, 2021.
M. Enzi, S.N. Parragh, & J.Puchinger.

Abstract The aim of the bi-objective multimodal car-sharing problem (BiO-MMCP) is to determine the optimal mode of transport assignment for trips and to schedule the routes of available cars and users whilst minimizing cost and maximizing user satisfaction. We investigate the BiO-MMCP from a user-centred point of view. As user satisfaction is a crucial aspect in shared mobility systems, we consider user preferences in a second objective. Users may choose and rank their preferred modes of transport for different times of the day. In this way we account for, e.g., different traffic conditions throughout the planning horizon.

We study different variants of the problem. In the base problem, the sequence of tasks a user has to fulfill is fixed in advance and travel times as well as preferences are constant over the planning horizon. In variant 2, time-dependent travel times and preferences are introduced. In variant 3, we examine the challenges when allowing additional routing decisions. Variant 4 integrates variants 2 and 3. For this last variant, we develop a branch-and-cut algorithm which is embedded in two bi-objective frameworks, namely the ϵ-constraint method and a weighting binary search method. Computational experiments show that the branch-and cut algorithm outperforms the MIP formulation and we discuss changing solutions along the Pareto frontier.

Keywords— car-sharing, mobility, transportation, bi-objective, branch-and-cut

4.1 Introduction

Today, most of the world's population lives in urban environments and cities continue to grow [144]. Urban mobility is therefore a key topic for a sustainable future. When considering a city's infrastructure, the available mobility offers are plentiful. Public transportation provides efficient connections, some commuters use their car, others prefer bikes, scooters or even taxi. Besides, a trend towards sharing is clearly visible in mobility (DriveNow, Uber, ...). In short, mobility as we use it and see it is changing.

This comes with a whole new stream of optimization problems. Only recently Mourad et al. [112] provided a survey on the vast topic of optimizing shared mobility.

The (privately owned) car is diminishing as the prevailing mode of transport in urban areas [145]. In Vienna (Austria), the number of cars per capita is constantly decreasing [105, 135]. People prefer other modes of transport (MOT). The modal split of cars shrank from 31% to 25% within the last decade. Within the same time period, bikes, public transportation and walking increased their modal split by 2 percentage points to 7%, 38%, and 30%, respectively [150, 151]. Thus, people move to alternative, more environmentally friendly MOTs.

Additionally, citizens increasingly use sharing systems [145]. In Germany, the number of shared cars has increased fivefold within ten years, there are almost twelve times more users than a decade ago [38]. In Vienna (Austria), one shared car eliminates the need of five privately owned ones [102]. At maximum 10% of the cars in Austrian households simultaneously drive on the roads. Many car owners use their vehicles only a couple of times per year. In Lisbon (Portugal), only 3% of the cars will be needed if all trips are covered by car- and ride-sharing. 95% parking space can be freed up [145]. Moreover, car-sharing saves up to 44 million car-kilometers in Vienna annually. This equals to approximately 7,000 tons of CO_2 [102]. Hence, by using car-sharing, resources can be employed more efficiently, it is more environmentally friendly, and newly available space can be gained as, e.g., green space in urban areas [145].

The importance of rethinking mobility is clearly visible in the presence of prominent concepts in various cities. Vienna targets a split of 80:20 where 20% of the trips are covered by cars, the others by public transportation, bikes or walking. The idea is to extend the mobility offers with profound sharing concepts and to move towards the vision of a 'Smart City' [133]. Madrid is aiming to establish a holistic 'Mobility as a Service' concept offering real-time information and including over 30 shared mobility options [46]. Within novel mobility concepts, bikes are receiving exceptional attention. Vienna almost doubled the cycling network in the last decade, and accomplished a similar increase in kilometers driven on specific legs [66, 103]. Paris presents the 'Plan Vélo' where the target is to emerge to the world's bike capital. The ambition is to minimize the space for cars and make space for bike usage and pedestrians [118].

Novel mobility concepts and reconsidering mobility plays an important role not only in a private environment, but also in a corporate setting. Companies increasingly aim to provide mobility concepts for their employees. This work is part of an applied research project SEAMLESS (http://www.seamless-project.at), in which project partners, such as the *Austrian Post AG* or *T-Systems Austria GesmbH*, strive for the implementation of the discussed ideas. The target is to reduce a one-to-one assignment of company cars, employ more environmentally friendly MOTs and strive for shared mobility where each employee gets her preference. This goes hand in hand with companies aiming for a greener carbon footprint and enhancing employee satisfaction [127].

Traveler experience needs to be taken into account in the design of novel mobility systems and is key to its success with users [5]. Thus, when studying mobility, convenience and user preferences are crucial. However, from an operator perspective the cost-factor plays an important role as well and usually cost-efficiency is in conflict with a MOT's convenience. This 'convenience' is difficult to measure and must be tackled on an individual user level. As we observe, and also other authors studying mobility have

outlined [67], including user preferences can decide on the 'win or lose' of a system. Therefore, we investigate the trade-off between minimizing cost and enhancing the individual satisfaction of a user in a mobility system. Combining these parameters and providing the decision maker with a set of efficient solutions will lead to an enhanced acceptance of such a system.

Motivated by this, we study the bi-objective multimodal car-sharing problem where we assign MOTs to trips and find car and, depending on the variant, also user routes throughout a day. We formulate two objectives to minimize cost and maximize user satisfaction. We further take into account the possibility of variation of user preferences and travel times throughout the day, becoming time dependent input parameters. We refer to car-sharing throughout the paper as to where a group of users is mutually using a pool of cars. Note that the output aims to provide an optimal assignment of MOTs throughout a day using time-dependent travel times.

Our main contributions are:

- We introduce the bi-objective multimodal car-sharing problem (BiO-MMCP). We present four variants of the problem, discussing increased flexibility of the timings of the visits: we present the model (i) with fixed task sequences and without time-dependent travel times and user preferences, (ii) with fixed time sequences and including time-dependent travel times and user preferences, (iii) no fixed sequences and no time-dependent travel times or preferences, and lastly (iv) open sequences of tasks and time-dependent travel times and user preferences.

- We propose a branch-and-cut algorithm for the most complex problem variant examined in this paper. The algorithm is embedded into two bi-objective frameworks, namely the ϵ-constraint method and a weighting binary search method. We show that for both frameworks it is highly beneficial to add cuts in the form of constraints from prior iterations to the following iterations.

- We provide a thorough analysis where we (i) compare different solution approaches for the models, and (ii) give insights into the trade-offs between cost-minimization and enhancing user-centred MOT preferences.

The paper is organized as follows: In Section 4.2 we review related work. Section 4.3 introduces the BiO-MMCP where Section 4.3.1 gives a problem formulation, followed by the formal description in Section 4.3.2 for all four variants. In Section 4.4 we describe the implemented solution approach. As most of the variants are solved as a mixed integer program (MIP) with the generic MIP solver CPLEX, we focus on the branch-and-cut developed for the last variant of the model, described in Section 4.4.2. Moreover, we introduce a set of valid inequalities in Section 4.4.1 and describe the bi-objective frameworks used in this paper in Section 4.4.3. Section 4.5 summarizes our computational study. Finally, we draw conclusions and we give an outlook to future work in Section 4.6.

4.2 Related work

Research addressing the design and implementation of car-sharing systems has risen over the past years. Many existing papers focus on strategic decision making, such

as the design of services, infrastructure (e.g. design/location of facilities or charging stations) or fleet management. Nevertheless, various papers stress the importance of integrating the user related attributes in optimization problems tackling sharing systems. A comprehensive literature review has been presented by Ferrero et al. [67].

A large amount of research has been performed on data collection, data analysis and simulation based studies in order to assess the potential impacts of car-sharing systems. Most of these studies have been conducted on city-wide public systems. Demand for car-sharing systems and impacts on mobility behaviour are typically assessed through questionnaires [155, 129]. The potential and effects of such systems are then often determined through simulation based approaches [45]. From an operational perspective problems considered in the car-sharing related literature are mainly concerned with relocating, recharging and servicing vehicles [51, 114, 149]. The problem we are introducing in this article however, is an operational problem for planning trips and allocating means of transport in a closed system where travel demand is known in advance. Embedding car-sharing in a multimodal system and especially treating it in a bi-objective formulation is a novel way of addressing car-sharing from a user-centered perspective.

In a different line of research, ride-sharing has attracted an increased amount of interest in the last years. Major research efforts have been made in analyzing and designing such services. Strategic and tactical decisions as well as the development of new algorithms for daily operations have also been in focus of recent work. A comprehensive survey on such approaches can be found in Mourad et al. [112]. A large number of case studies mainly based on simulation and data analysis have been published on the potential impact and feasibility of various sharing schemes with a focus on ride-sharing [41, 56, 104, 137].

For the first two variants of our proposed problem, where the task sequence is fixed, we refer to the Fixed Sequence Arc Selection Problem (FSASP) which was introduced by Garaix et al. [69] and proven to be NP-hard. The FSASP considers a fixed sequence of nodes that are linked by multiple arcs. Choosing an arc between two nodes is the subject of determination. This problem applies to the first two variants of our problem in this paper. Note that only recently Huang et al. [87] shortly stressed that this research direction can be a good basis for further algorithmic work, naming home appliance delivery companies as an example. As we additionally determine the sequence of visited nodes, we can detect similarities to the VRP [141, 60] in our work. Our paper introduces a kind of multi-trip VRP [43] with heterogeneous vehicles and multiple depots on a multi graph. Garaix et al. [69] were among the first who studied VRPs with alternative arcs between each pair of nodes. VRPs with multiple attributes [69] or multi-graphs in the VRP stream have gained increasing attention in the past years [55, 20, 21, 22, 87], whereas, of course, mutlimodality significantly enlarges the set of possible solutions [42]. Research considering various attributes on arcs is fairly recent, yet highly important to consider as one connection of nodes usually implies specific trade-offs (usually time vs. cost) which are not considered on a classical graph. We consider the characteristics of different modes of transport as well as time-dependent preferences and costs jointly on one arc. We refer to Gendreau et al. [70], for a review on time-dependent routing problems. However, we could not find any work introducing time-dependent preferences on modes of transport in a car-sharing context.

Integrating customer-oriented aspects into optimization problems, or more specific vehicle routing problems, is a topic of increasing interest. In Vidal et al. [146] a detailed analysis through VRP variants also tackling customer-centred objectives is provided. As an example, the cumulative VRP [115, 128] replaces the classical minimum cost objective function with the minimization of individual customer arrival times. Martínez-Salazar et al. [106] introduce a customer-centric multi-trip VRP with a single vehicle minimizing the sum of customer waiting times to receive a specific service. On a somewhat different but related topic, Braekers et al. [33] introduce a bi-objective routing and scheduling problem for home care where the second objective minimizes client inconvenience. In our work, we optimize user preferences for MOTs as a second objective function. Jozefowiez et al. [89] review numerous papers tackling multiple objectives in the context of VRPs. They name the most common objectives to be cost, length of the tour, balance or problem specific objectives. Since then, various papers have been published. Recently it seems that there is a vast amount of published research with environmental [1, 6, 8, 53, 64, 71, 140, 143, 76, 7, 77] or external social criteria [71, 76, 116, 7, 77] as alternative objectives.

Multi-objective optimization gives a deeper insight into the solution pool of a problem. However, there might exist a large number of trade-off solutions. The target is to find an efficient set of solutions that cannot be optimized in one objective without worsening another one. Those efficient solutions are then called Pareto optimal solutions. There is a vast amount of works on exact as well as heuristic approaches to solve for multicriteria optimization problems. Prevailing metaheuristics in this field are evolutionary algorithms such as the NSGA-II [52] or the SPEA-II [72]. However, only recently Matl et al. [109] have shown that single-objective VRP heuristics can be efficiently used in an ϵ-constrained-based method. The ϵ-constraint method [153, 131] is one of the prevailing methods to solve multi-objective optimization problems. It repeatedly solves a single-objective optimization problem by considering the other objectives in terms of constraints. Further widely applied frameworks to solve multi-objective problems are the two-phase method [148], the weighted sum approach [9] or, more recently, the balanced box method [30] and the weighting binary search method [126]. These so called criterion space methods, embed a single-objective optimization problem and systematically enumerate the Pareto frontier. However, recent works focus on adapting the branch-and-bound algorithm to solve the multi-objective case in a single run [136, 147, 119, 2]. A recent overview of exact methods for multi-objective optimization is provided in Ehrgott et al. [59]. A detailed overview of general multi-objective combinatorial optimization is provided by Ehrgott and Gandibleux [58]. For our study we choose the ϵ-constraint method as well as a weighting binary search as they are relatively simple to implement and have shown to be very efficient. The latter one is based on the algorithm proposed by Riera-Ledesma and Salazar-González [126], who developed a weighting method and conduct a binary search in the objective space. Moreover, similar to Bérubé et al. [25], we use a branch-and-cut approach relying on previous information for subsequent problems by adding cuts to the subproblem. Similarly in Riera-Ledesma and Salazar-González [126] cuts from prior iterations are added to the cut pool for further iterations. Contrary to Riera-Ledesma and Salazar-González [126] and Bérubé et al. [25], we add detected cuts as hard constraints, showing better results for our problem setting.

4.3 The bi-objective multimodal car-sharing problem

In the following we describe the BiO-MMCP and give a formal description of the variants of the problem studied in this paper.

4.3.1 Problem description

The BiO-MMCP aims to assign modes of transport to user trips and determining car routes during a day while minimizing cost and maximizing user satisfaction by accounting for MOT preferences.

Each user trip starts in a depot, covers a set of tasks and ends in a depot again. A user may have more than one trip during a day. A route is a sequence of trips during a day. Note that we introduce car routes and user routes: A car route schedules the trips covered by one car during a day, whereas the car is handed over at the depot from one user to another. A user route consists of all the trips assigned to one user during a day, whereas the user may change MOTs between trips (i.e. at the depot).

We consider a closed group of users and a set of possible MOTs. A pool of cars is given and all other MOTs are considered to have infinite capacity. With this assumption we are able to cover all demanded trips. This also has practical implications as, e.g., there is usually no spatial or temporal limit on the availability of public transport in a city during a day. This also holds for bikes, as due to several bike-sharing offers, we can assume that bikes are available at any time in a city. Each user may give preference scores to the available MOTs where we assume the lower the score the better the MOT is rated (scale 1-10 where 1 is best). Moreover, depending on the variant of the problem, users may determine preferences for different times of the day, resulting in time-dependent user-based MOT preferences. Furthermore, we introduce time-dependent travel times as, e.g., the car drive will take longer through rush-hour than at noon. As our cost function also comprises cost of time, the adapted travel times will have an impact on the cost function. Note that even though travel times may be stochastic, we can plan within a deterministic setting as we use time-dependent travel times for all modes of transport.

The goal of the BiO-MMCP is to cover a set of trips for a given planning horizon by assigning MOTs to trips and determine car routes (optionally also user routes) for a closed community. The locations of the start and end points as well as the tasks of a trip are fixed. This means, it is known in advance which user will visit which task. Depending on the considered variant of the problem, the sequence of the tasks may vary.

We investigate four variants of the introduced problem:

Model 1 (`m1`) In the first variant we assume that each user follows a fixed sequence of tasks, starting and ending at a fixed (but possibly different) depot. Preferences are given for each MOT for each user. We aim to find the best MOT to trip assignment and to determine the car routes. The objectives are to minimize costs and MOT preferences. In this variant, user routes are assumed to be given.

Model 2 (`m2`) In this variant we assume the same setting as in model `m1` but include time-dependent MOT preferences and travel times. The target is to find the best MOT

to trip assignment and schedule the car routes from a pool of cars whilst minimizing time-dependent costs and user preferences. Again, user routes are input to the problem.

Model 3 (`m3`) In the third variant we consider a fixed user to tasks assignment, and start and end locations. However, the sequence of tasks within a trip as well as the sequence of user trips are subject of determination. This means that we have to, in addition to car routes, find user routes throughout a day. The objectives are again to minimize costs and user preferences.

Model 4 (`m4`) This model is a combination of model `m2` and `m3`: we consider time-dependent user preferences and travel times as well as variable task and trip sequences. Thus, we intend to determine the MOT assignment, schedule car as well as user routes whilst minimizing both time-dependent MOT preferences of users and costs.

4.3.2 Formal description

We now formally introduce the different variants and their respective mathematical formulations, using the following notation (also summarized in Table 4.1):

Given is a set of users P and a set of trips R, where each trip $r \in R$ has a set of tasks Q_r assigned. A trip starts in a depot a_r, ends in depot b_r and covers in between one or more tasks q. We store all nodes assigned to a trip r in the set G_r, where $r = \{a_r, q_1^r, q_2^r, ..., b_r\}$. Note that a user p might cover more than one trip during a day. The set of tasks Q_r is known in advance whereas each task q is unique and may only be in one set $Q_r \subseteq Q$, where Q denotes the set of all tasks. We model the connections between two subsequent tasks as a leg l.

Furthermore, we consider a set of depots D, which are artificial nodes representing start/end points of car routes during a day, i.e. each route starts and ends here. The start depot d is connected to all starting nodes a, and conversely each end node b is connected to the end depot d'.

We consider a set of modes of transport $K = \{car, public, bike\}$, where *public* comprises public transportation including walking. If a trip starts by a MOT, then the MOT will be used for the full trip. We assume at each depot $d \in D$ an available number of MOTs k at the beginning and end of the planning horizon, denoted as W_{dk} and $\overline{W}_{d'k}$, respectively.

We denote the set of all nodes by V and V' be the set of nodes without the set D, such that $V' = V \setminus D$. For every node $v \in V$ we have the set of outgoing legs L^+_{vk} and ingoing legs L^-_{vk} by MOT k. All legs are stored in the set of all legs L. We store any relevant information on the legs.

Each user p assigns a preference value σ_{pk} to each of the given modes of transport $k \in K$. Note that, as we also minimize the preference objective, we assume that the lower the score, the better the user values the mode of transport. As a leg l refers to exactly one mode of transport k and one user p, we assign the value σ_{pk} to the respective leg l, denoted as θ_l. The cost value c_l of a leg l consists of variable distance cost, cost of time and cost of emissions. For more information, we refer to Section 4.5.1.

For time-dependent user preferences we define a set of time periods $t \in T$ during the day. A time period replicates, e.g., rush-hours. Each user p determines a preference value σ^t_{pk} for each of the given time periods t and MOT k. In the case when a leg l

Table 4.1: Mathematical notation used in the formal description of the BiO-MMCP.

Sets and nodes

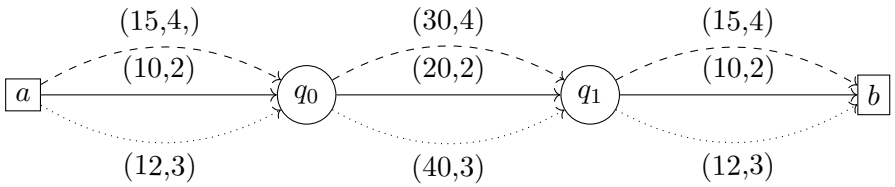
P	set of users
R	set of trips
$Q_r \subseteq Q$	set of tasks of trip r as a subset of the set of tasks
a_r	start node of a trip r
b_r	end node of a trip r
G_r	set of nodes on a trip r
D	set of depots
K	set of modes of transport
L	set of legs
L^-, L^+	set of ingoing/outgoing legs
L_v^-, L_v^+	set of ingoing/outgoing legs of node v
L_d^-, L_d^+	set of ingoing/outgoing legs of depot d
L_p	set of legs assigned to user p
L_r	set of legs on a trip r
L_{vp}	set of legs of a user p going in/out of a node v
L_{vk}^-, L_{vk}^+	set of ingoing/outgoing legs of node v by MOT k
L_{vp}^-, L_{vp}^+	set of ingoing/outgoing legs of node v by user p
T	set of time periods
S	subset of the set of nodes G_r of a trip r
$A_p \subseteq A$	set of trip start nodes of a user p as a subset of the set of trip start nodes
$B_p \subseteq B$	set of trip end nodes of a user p as a subset of the set of trip end nodes
V	set of all nodes
V'	set of nodes without depots, V \ D
T	set of time periods
$\mathcal{R}$	set of infeasible paths
$\mathcal{R}_{car}$	set of infeasible car routes
$\mathcal{R}_p$	set of infeasible user routes
γ_p	start node of person p
ϕ_p	end node of person p

Input parameter

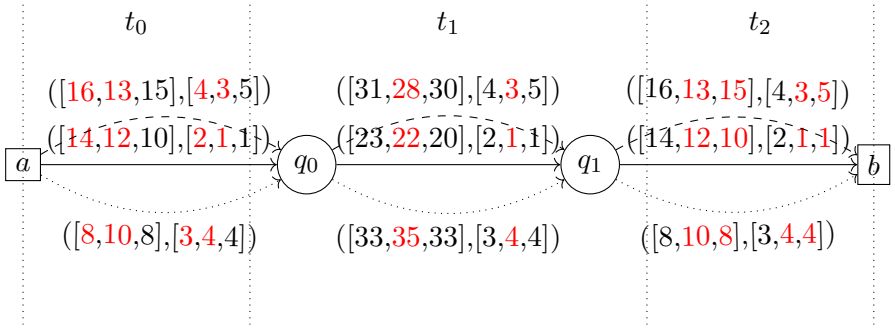
$W_{dk}, \overline{W}_{dk}$	number of MOTs k in depot d at beginning/ end of the planning horizon
$\sigma_{pk}, \sigma_{pk}^t$	preference value of a person p for MOT k (for time period t)
θ_l	preference of leg l
y_l	origin of leg l
z_l	end of leg l
c_l	cost of leg l
u_l	user of leg l
m_l	MOT of leg l
h	maximal waiting time
H	end of planning horizon
M	big M
t_l	driving time of leg l
s_v	service time at node v
o_l	interval start of leg l
e_l	interval end of leg l
$\mathcal{W}$	accumulated waiting time
Δ	value stating how much a route can be moved forward
F	forward slack, $F = \mathcal{W} + \Delta$

Decision variables

x_l	1 if leg l is chosen, 0 otherwise
τ_l	departure time of leg l



(a) Trip r with its associated legs l, and the respective cost and preference values given as (c_l,θ_l).



(b) Trip r with its associated legs l, and the respective time-dependent cost and user-preferences for each time period t given as $([c_l^{t_0},c_l^{t_1},c_l^{t_2}]),[\theta_l^{t_0},\theta_l^{t_1},\theta_l^{t_2}]$.

car public transp. bike

Figure 4.1: Example of one trip with its associated legs l starting in node a, visiting tasks q_0, q_1 and ending in node b. Between the nodes we insert different legs for each mode of transport, which are car, public transportation and bike in our case. A label of a leg is defined with two attributes: cost and preferences. Figure (a) shows a simple trip, where no time-dependencies are considered. Figure (b) includes time-dependent information for the respective periods t.

completely lies within a period t the preference value of the leg θ_l equals σ_{pk}^t. In the case where the leg covers more than one period, we calculate a weighted average of the preference values. As our cost also depends on time, we also adapt the cost term considering time-dependencies in the same way.

Figure 4.1(a) shows an example of a simple trip r. It starts in node a and ends in b whilst visiting q_0 and q_1. We insert legs for each mode of transport (denoted by different lines) between each node and assign the respective cost and preference value, given in brackets as (c_l,θ_l). We do not consider time-dependent travel times or user preferences here. Figure 4.1(b) shows the same trip as Figure 4.1(a), but considers time-dependencies. Therefore, three time periods are indicated as t_0,t_1,t_2. For each leg we have cost and preference values for each of the respective periods. The legs between q_0 and q_1 lie completely within one time period and can therefore be taken as they are. For the others, we compute the share of each time period on the leg and get the respective preference value and cost by computing the weighted average.

Model 1 (`m1`)

In model `m1` the sequence of tasks is fixed, resulting in predetermined trips $r \in R$. We connect each a with its fixed successor q, each task q with its fixed successor q' or,

if the trip only covers one task, the trip end node b. For every pair of end and start nodes (b, a) where a is ahead in time, we insert an additional artificial leg with costs and preferences 0, in order to allow for the connection of car routes covering more than one trip throughout the day.

Each leg in the graph results in a tuple $\{(u_l, y_l, z_l, m_l, c_l, \theta_l)\}$ where u_l is the assigned user, y_l and z_l are the origin and end of the leg, m_l the assigned MOT, c_l the cost and θ_l the preference value.

The introduced binary decision variable x_l takes on value 1 if leg l is chosen and 0 otherwise.

With this, we can introduce a compact formulation for the first version of the BiO-MMCP.

$$\min \quad \sum_{l \in L} c_l x_l \tag{4.1}$$

$$\min \quad \sum_{l \in L} \theta_l x_l \tag{4.2}$$

$$\text{s.t} \quad \sum_{k \in K} \sum_{l \in L^+_{vk}} x_l = 1 \qquad \forall v \in V' \tag{4.3}$$

$$\sum_{l \in L^-_{vk}} x_l = \sum_{l \in L^+_{vk}} x_l \qquad \forall v \in V', k \in K \tag{4.4}$$

$$\sum_{l \in L^+_{dk}} x_l \leq W_{dk} \qquad \forall d \in D, k \in K \tag{4.5}$$

$$\sum_{l \in L^-_{d'k}} x_l \leq \overline{W}_{d'k} \qquad \forall d' \in D, k \in K \tag{4.6}$$

$$x_l \in \{0, 1\} \qquad \forall l \in L \tag{4.7}$$

The objective (4.1) minimizes total cost and objective (4.2) minimizes user-centred MOT preferences. Constraints (4.3) make sure that each node v is covered by exactly one leg l. Constraints (4.4) ensure flow conservation at nodes $v \in V'$ for every MOT k. Constraints (4.5) and (4.6) restrict the number of available MOTs $W_{dk}, \overline{W}_{d'k}$ at the start and end of the time horizon. Constraints (4.7) define the domains of the decision variables.

Model 2 (`m2`)

We extend the previous model by introducing time-dependent MOT preferences and costs. We assume fixed times of tasks q. With this, and as we know the driving time of a leg, we can exactly determine start and end times of the leg and thus assign a preference value.

As we store all relevant information directly on the leg l, we do not have to model time explicitly. This results in the same tuple $\{(u_l, y_l, z_l, m_l, c_l, \theta_l)\}$ as before, with a modified value of θ_l and c_l. As we only have a change in the data, but the model remains unchanged, we use model `m1` again.

Model 3 (`m3`)

In model `m3` we have to determine the sequence of tasks per user (ensuring no subtours) as well as consider the scheduling of trips each user is taking. Therefore, the underlying graph has to be adapted. We again consider the set of all nodes V, the set of intermediate nodes V', the set of depots D, the set of MOTs K, the set of legs L, and the set of users P. We define sets A_p and B_p containing all start nodes a and end nodes b of a user p, respectively. These sets will consist of exactly one node, if a user is taking only one trip, two if the user has two trips, etc. Previously, to assure car routes, we only connected an end node b of a trip to a start node a of another trip if a was ahead in time of b. As we are not considering any fixed times/sequences anymore, we connect every b to every a if they are in the same physical depot. Similarly we connect all nodes belonging to one set G_r, yet not changing the predetermined start and end nodes of one trip. For now, we do not consider time-dependent preferences on legs. Note that the tasks lying on a specific trip are fixed, meaning that if a user previously had two trips, the user will again cover two trips.

In order to prevent parallel trips of one user, the user routes are modeled into the graph. Doing so, we add new artificial nodes γ_p and ϕ_p for each user p where the user starts and ends the respective route during a day (similar to the idea of the $d \in D$ where all MOT flows start). We connect the respective γ_p to all start nodes $a \in A_p$ of one user p and conversely the respective ϕ_p to all $b \in B_p$. We connect user trips by inserting a leg l between b, a of the same user. Note that, instead of modifying the underlying graph, we also used additional constraints in the model. However, this formulation turned out to be very weak.

As the sequence of tasks of a trip is not fixed we determine the departure time τ_l of a leg l. By assuring increasing times of legs, we also avoid subtours. Additionally, in order to avoid unrealistic long waiting times at nodes, we assume that a user can wait for a maximum amount of time before she continues her trip, e.g. 30 minutes, denoted as h.

Model `m3` can now be stated as follows, where decision variables τ_l give the departure time of leg l, H depicts the end of the planning horizon, M denotes a big M, t_l is the travel time of a leg l and s_v the duration of the task.

$$\begin{aligned}
\min \quad & (4.1) \\
\min \quad & (4.2) \\
\text{s.t} \quad & (4.3) - (4.7) \\
& \tau_l + t_l + s_v - \tau_n \leq M(2 - x_l - x_n) && \forall l \in L^-_{vk}, n \in L^+_{vk}, \\
& && \quad v \in V', k \in K && (4.8) \\
& \sum_{l \in L^-_v} (\tau_l + t_l x_l) + s_v \geq \sum_{l \in L^+_v} \tau_l - h && \forall v \in V' && (4.9) \\
& \tau_l \leq H x_l && \forall l \in L && (4.10) \\
& \sum_{l \in L^+_{\gamma_p}} x_l = 1 && \forall p \in P && (4.11) \\
& \sum_{l \in L^-_{\phi_p}} x_l = 1 && \forall p \in P && (4.12)
\end{aligned}$$

$$\sum_{l \in L_{vp}^-} x_l = \sum_{l \in L_v^+} x_l \qquad \forall v \in A_p, p \in P \tag{4.13}$$

$$\sum_{l \in L_v^-} x_l = \sum_{l \in L_{vp}^+} x_l \qquad \forall v \in B_p, p \in P \tag{4.14}$$

$$\tau_l + t_l + s_v - \tau_n \leq M(2 - x_l - x_n) \quad \forall l \in L_{vp}^-, n \in L_{vp}^+, v \in V' \cup \{\gamma_p, \phi_p\}, p \in P \tag{4.15}$$

$$\tau_l \geq 0 \qquad \forall l \in L \tag{4.16}$$

Constraints (4.8) set the time variables and take care of subtour elimination within trips. Constraints (4.9) ensure that a user is leaving at the latest h minutes after the end of the task. Constraints (4.10) restrict the latest departure time at any task to be at the end of the time horizon. Constraints (4.11) and (4.12) make sure that each user is starting her route in node γ_p and ending in node ϕ_p. Constraints (4.13) and (4.14) balance the flows of start and end nodes of user p. Constraints (4.15) eliminate parallel trips. Finally, constraints (4.16) make sure that decision variables τ are non-negative.

Model 4 (`m4`)

Lastly, in addition to a flexible sequence of tasks, in model `m4` we add time-dependent MOT preferences to the model. This is mainly done by adapting the graph and by adding one constraint to the model `m3`.

We discretize time in intervals of α minutes and duplicate each leg $l \in L$ for each interval. Note that time-dependent MOT preferences are derived from the user preference values σ_{pk}^t.

We extend the leg information by adding the start and end times of the interval lying on the leg, this results in the tuple $\{(u_l, y_l, z_l, m_l, c_l, \theta_l, o_l, e_l)\}$ where o_l gives the start time and e_l the respective end time of the interval.

Finally, we append the following constraints to model `m3`:

$$o_l x_l \leq \tau_l \leq e_l \quad \forall l \in L \tag{4.17}$$

Constraints (4.17) make sure that τ_l of leg l lies within the predetermined times.

The resulting model relies on both binary and continuous variables. We adapt this and use a re-formulation that is of exponential size but relies on binary variables only. We replace constraints (4.8), (4.9), (4.10), (4.15), (4.16), and (4.17) by infeasible path constraints [11] (for car routes and user routes), and subtour elimination constraints.

Let $\mathcal{R}_{car}$ denote the set of infeasible car routes, and $\mathcal{R}_p$ be the set of infeasible user routes. $V(S)$ gives the nodes of the set S, where S is a subset of the set of nodes on a trip G_r. Legs of an infeasible path ρ are denoted as $L(\rho)$. Model `m4b` can be stated as follows:

$$\begin{aligned}
&\min && (4.1)\\
&\min && (4.2)\\
&\text{s.t} && (4.3)-(4.7),(4.11)-(4.14)\\
&&& \sum_{l\in L(\rho)} x_l \leq |L(\rho)|-1 && \forall \rho \in \mathcal{R}_{car} && (4.18)\\
&&& \sum_{l\in L(\rho)} x_l \leq |L(\rho)|-1 && \forall \rho \in \mathcal{R}_{p} && (4.19)\\
&&& \sum_{l\in L(S)} x_l \leq |S|-1 && \forall S \subseteq G_r, r \in R, S \neq \emptyset && (4.20)
\end{aligned}$$

Constraints (4.18)-(4.19) eliminate the infeasible paths of cars and users. We sum over all legs l of the respective infeasible path ρ, and set it infeasible by denoting that at least one leg cannot be on the route. Constraints (4.20) are subtour elimination constraints. We set the constraints for all trips r where we store the nodes of each trip in the set G_r.

4.4 Solution approach

In the following, we first introduce valid inequalities in Section 4.4.1. By embedding the models into bi-objective optimization frameworks, described in Section 4.4.3, the scalarized models `m1`, `m2` and `m3` are solved with CPLEX. We can solve real-world sized instances within seconds, as we will show in our computational results. However, as expected, `m4` is more challenging to solve. Therefore, we develop a branch-and-cut algorithm in Section 4.4.2 for model `m4b`.

4.4.1 Valid inequalities

In order to strengthen the models `m3`, `m4`, and `m4b`, the following set of valid inequalities is used.
We know that all legs of a trip must be covered by a single MOT. Therefore, we can say that either MOT k is going into node v, or any other MOT $g \neq k$ out of a node v, but not both. Assuming that the ingoing legs of a node v are stored in the set L^-_{vg} and all outgoing legs of a node v are stored in the set L^+_{vk}, we can state:

$$\sum_{l\in L^+_{vk}} x_l + \sum_{g\in K: g\neq k} \sum_{l\in L^-_{vg}} x_l = 1 \quad \forall v \in V', k \in K \tag{4.21}$$

In `m3`, `m4`, and `m4b`, we only require that the sum over all outgoing legs of a node must be equal to 1. In the following valid inequality the sum over all ingoing legs $l \in L^-_{vk}$ using MOTs k of a node v has to be equal to 1:

$$\sum_{k\in K} \sum_{l\in L^-_{vk}} x_l = 1 \quad \forall v \in V' \tag{4.22}$$

Since a car may cover more than one trip, but has to take at least one if it departs from the depot d, the number of trips started with a car (leaving from any node $a \in A$) has to be greater or equal to the sum of cars leaving any depot $d \in D$. Ingoing legs of the start nodes a using MOT k are given in the set L^-_{ak}, outgoing legs of the depot d are given in the set L^+_{dk}. The constraint is valid for cars only. Thus, we sum over all the ingoing legs of any node a, which then has to be greater or equal to the sum over all outgoing legs of any depot d:

$$\sum_{a \in A} \sum_{l \in L^-_{ak}} x_l \geq \sum_{d \in D} \sum_{g \in L^+_{dk}} x_g \quad \text{with k = car} \tag{4.23}$$

Assuming that a user p has been assigned a single task only, then a full user route will be: $\gamma_p - a_p - q - b_p - \phi_p$. This means, the shortest possible user route consists of four legs. Assuming that all legs assigned to a user p are stored in the set L_p, we can formulate:

$$\sum_{l \in L_p} x_l \geq 4 \quad \forall p \in P \tag{4.24}$$

Assuming that a trip has at least one task, then each trip will consist of at least three nodes $(a - q - b)$, and thus two legs. The sum over all legs of a trip r is at least the number of nodes assigned to the respective trip, given in the set G_r, minus 1. :

$$\sum_{l \in L^-_v : v \in G_r} x_l \geq |G_r| - 1 \quad \forall r \in R \tag{4.25}$$

As we know the number of tasks a person is covering, we also know the number of legs the person will cover in the solution. Therefore, we can introduce the following constraint where L_p is the set of legs of a person p and V_p gives the nodes assigned to person p:

$$\sum_{l \in L_p} x_l = |V_p| - 1 \quad \forall p \in P \tag{4.26}$$

We add cycle constraints, meaning that we can only go either from v to v' or from v' to v, but not both. We store all legs that start in v and end in v' in the set $L^{(v,v')}$ and vice versa in $L^{(v',v)}$. With this we formulate the following valid inequality:

$$\sum_{l \in L^{(v,v')}} x_l + \sum_{l \in L^{(v',v)}} x_l \leq 1 \quad \forall (v, v') \in L \tag{4.27}$$

The above valid inequalities are used to strengthen `m3`, `m4`, and `m4b`. We now propose additional valid inequalities which are used to strengthen only `m4` and `m4b`.

Let us consider a node v, a leg l leaving the node v, and an ingoing leg g. As described in Section 3.2.4, for the time-dependent setting of the model, the legs contain intervals with the possible start and end time information (o, e). With this, we know that the

start and end times of the outgoing leg l has to be greater than the times of the ingoing leg g. Therefore, if the start and end times of the ingoing leg g are greater than the times of the outgoing leg l, meaning that the ingoing leg would happen later in time, only one of them can be used:

$$o_l < o_g \wedge e_l < e_g \iff x_l + x_g \leq 1 \quad \forall l \in L_v^+, g \in L_v^-, v \in V' \tag{4.28}$$

As any outgoing leg of a node v has to be later than the ingoing leg of the respective node, we can further eliminate all outgoing legs of a node v that are timed before a chosen ingoing leg of the respective node. Therefore, we adapt equation (4.28), where we assume an ingoing leg $g \in L_v^-$ of a node v and sum over all outgoing legs $l \in L_v^+$ with smaller start and end times as the ingoing leg, thus $o_l < o_g$ and $e_l < e_g$. Then at most one of the respective legs can be chosen. Conversely, considering an outgoing leg l and summing over all ingoing legs $g \in L_v^-$ with an interval greater than the one of the outgoing leg ($o_g > o_l$,$e_g > e_l$), we can again say that at most one leg can be chosen. Both valid inequalities can be formulated as follows:

$$x_g + \sum_{l \in L_v^+ : o_l < o_g \wedge e_l < e_g} x_l \leq 1 \quad \forall g \in L_v^-, v \in V' \tag{4.29}$$

$$x_l + \sum_{g \in L_v^- : o_g > o_l \wedge e_g > e_l} x_g \leq 1 \quad \forall l \in L_v^+, v \in V' \tag{4.30}$$

If the beginning of the interval o_l of the outgoing leg l is greater than the end of the interval e_g of the ingoing leg g plus the time of the ingoing leg t_g plus the service time at the node s_v plus the maximum waiting time h, then these legs are not compatible in time. Again, considering a node v with outgoing legs L_v^+ and ingoing legs L_v^-, we can state the following valid inequalities:

$$x_g + \sum_{l \in L_v^+ : o_l > e_g + t_g + s_v + h} x_l \leq 1 \quad \forall g \in L_v^-, v \in V' \tag{4.31}$$

$$x_l + \sum_{g \in L_v^- : o_l > e_g + t_g + s_v + h} x_g \leq 1 \quad \forall l \in L_v^+, v \in V' \tag{4.32}$$

If the beginning interval o_g of the ingoing leg g plus the travel time of the ingoing leg t_g plus the service time of the node s_v is greater than the end of the interval e_l of the outgoing leg l, then these legs cannot be used together. We can again put this into two valid inequaities as follows:

$$x_g + \sum_{l \in L_v^+ : o_g + t_g + s_v > e_l} x_l \leq 1 \quad \forall g \in L_v^-, v \in V' \tag{4.33}$$

$$x_l + \sum_{g \in L_v^- : o_g + t_g + s_v > e_l} x_g \leq 1 \quad \forall l \in L_v^+, v \in V' \tag{4.34}$$

4.4.2 Branch-and-cut for `m4b`

In order to solve model `m4b`, we develop a branch-and-cut algorithm. Branch-and-cut algorithms make use of a subset of constraints and iteratively add further constraints in a cutting-plane fashion. Usually, constraint sets of exponential size are excluded which reduces the model to a reasonable size. In our case, we separate the infeasible path constraints (4.18)-(4.19) but we enumerate all subtour elimination constraints, since trips are usually very short. Separation algorithms are then called to determine whether the current solution is feasible by checking the omitted constraints. Note that the separation algorithms can be called on any relaxed solution or only on incumbent ones. Our strategy is based on the latter case, where we only call the algorithms if a new incumbent solution is found. If the separation algorithm detects a violation, the respective constraint is added as a cut to the model and the model is consecutively resolved. This is repeated until no violating constraints are detected and an optimal solution is found.

In our model a route (path) may be infeasible due to (i) user related constraints, (ii) shared cars related constraints, and (iii) synchronization requirements between user and car routes. Therefore, we first check if all user routes are feasible, then if all car routes are feasible and finally if they are both simultaneously feasible. The respective separation procedures are described in the following.

Separation of infeasible user routes

We separate infeasible user routes for each user $p \in P$. Let $\overline{x}$ denote the solution at the current node in the branch-and-bound tree. We start the construction of the route ρ at node γ_p. We denote the currently considered node as node v. From the starting point, we append the outgoing leg l at node v (v.outgoing) if $\overline{x}_l = 1$ to the route ρ, and update v to be the end node of leg l (l.endNode). We do this until we hit the user end depot ϕ_p.

In the following, we consider a forward slack F, consisting of an accumulated waiting time $\mathcal{W}$ and a value stating how much we could move the whole route such that the solution would still be feasible, given as Δ, and $F = \mathcal{W} + \Delta$. The current time stamp is given as τ. Before checking the route ρ for time feasibility, we initialize F, $\mathcal{W}$, τ to 0, and $\Delta = \infty$.

We iterate through the route as long as all time constraints are respected. We start by checking the second leg l on route ρ and systematically take the consecutive one. Thus, considering the current leg l leaving node v, we set $\tau = \tau + s_v + t_{l-1}$ and update $\mathcal{W}$ and Δ. The accumulated waiting time is calculated as the current waiting time plus either the maximum possible waiting time h or the remaining time to the end of the interval e_l, such that $\mathcal{W} = \mathcal{W} + min\{max\{0, e_l - \tau\}, h\}$. We can further push the whole route to the end of the given interval e_l or by the previously stored Δ. We update $\Delta = min\{\Delta, max\{0, e_l - (\tau + \mathcal{W})\}\}$ and compose $F = \mathcal{W} + \Delta$. If the current time τ lies within the respective interval of leg l (o_l, e_l), we can proceed to the next leg. If not, we try to push the route to the starting interval o_l of the current leg l, but at maximum by adding F, such that $\tau = \tau + min\{max\{0, o_l - \tau\}, F\}$. If the adapted τ violates the timing restrictions, the corresponding infeasible path constraint is generated. If τ is feasible $(o_l \leq \tau < e_l)$, we can update $\mathcal{W}$ and Δ, and proceed with the next leg. To update the values, we have to deduct the respective time used up of the forward slack. For this, we

first adapt Δ by stating $\Delta = \Delta + min\{\mathcal{W} - (\tau - \tau'), 0\}$, where τ' denotes the time stamp before adding the time slack F. The waiting time is updated as $\mathcal{W} = max\{\mathcal{W} - (\tau - \tau'), 0\}$. The pseudocode is outlined in the Appendix A.3 in Algorithm 1.

Separation of infeasible car routes

We further aim to detect infeasible paths regarding cars violating time constraints. We adopt the same idea as above, except following car routes. Starting depots of cars are $d \in D$. Note that, as we might have more than one trip originating from one node d, we slightly adapt the construction of the route ρ by considering node d multiple times as a starting node for the construction of the route ρ. We store the outgoing leg l of node v with $\overline{x}_l = 1$ and the MOT $k = car$ in the route ρ. While constructing, we save the number of trips on the current route, as we only consider routes with more than one trip. Timing restrictions for a single trip are already covered in the user route separation. If the route ρ consists of multiple trips, we follow the same steps as previously described in the separation algorithm of user routes. The pseudocode is given in Algorithm 2 in the Appendix A.3.

Synchronization of routes

It is not sufficient to check user and car routes separately for infeasibility. We also have to check if the user and car routes are synchronized, i.e. if the user who has taken over a car is at the depot at the respective time. In order to do so, we consider the whole solution and we store the used legs in the subset L', and obtain the sets $L'^{-}_{vk}, L'^{+}_{vk}, L'^{-}_{vp}, L'^{+}_{vp}$ ($\overline{x}_l = 1$ in the current solution), and we solve the following small LP derived from constraints (4.8), (4.9) and (4.15):

$$\tau_l + t_l + s_v \leq \tau_n \quad \forall l \in L'^{-}_{vk}, n \in L'^{+}_{vk}, v \in V', k \in K \tag{4.35}$$

$$\tau_l + t_l + s_v \leq \tau_n \quad \forall l \in L'^{-}_{vp}, n \in L'^{+}_{vp}, v \in V' \cup U, p \in P \tag{4.36}$$

$$\tau_l + t_l + s_v \geq \tau_n - h \quad \forall l \in L'^{-}_{vk}, n \in L'^{+}_{vk}, v \in V', k \in K \tag{4.37}$$

Constraints (4.35) and (4.36) synchronize the car and user route with the decision variable τ. Furthermore constraints (4.37) make sure that the waiting time at a node is not exceeded.

The above constraints are infeasible if the respective user and car are not at the same time at the same place. Therefore we can assume that either a car trip or an arc connecting the user trips is infeasible. Thus, we insert into the set L'' all legs from the set L' that are taken by car or are connecting user trips in the current incumbent solution, and we add the following constraint:

$$\sum_{l \in L''} x_l \leq |L''| - 1 \tag{4.38}$$

Strengthened infeasible path constraints

The infeasible paths introduced before in the form of constraint (4.18) and constraint (4.19) are very weak. We strengthen them as follows:

$$\sum_{l\in\rho} x_l + \sum_{l\in\mathcal{L}'} x_l + \sum_{l\in\mathcal{L}''} x_l \leq |\rho| - 1 \tag{4.39}$$

Let $\mathcal{L}'$ contain all legs with the same start node y, end node z and MOT k but earlier or later intervals (o, e) than the last checked leg of the separation algorithm, i.e., where the infeasiblity was detected. Let l' be the last checked leg, and τ the current departure time. If $\tau > e_{l'}$, meaning that we have jumped over the interval, then the set $\mathcal{L}'$ contains all legs with the same respective y, z, k but $o < o_{l'}$. This means that if we missed the interval, then also all prior ones will be too early. Conversely, if the interval of leg l' could not be reached, thus $\tau < o_{l'}$, we put all legs with the same y, z, k as leg l' but $o > o_{l'}$ into the set $\mathcal{L}'$. Hence, if we were not able to reach the respective interval, then also all later legs will not be reachable.

The set $\mathcal{L}''$ also depends on whether we are not able to reach the leg's interval or we miss it. We consider all legs on the route ρ except the last checked leg l', denoted as ρ'. Considering $\tau < o_{l'}$, thus the time stamp lies before the start of the interval, then the set $\mathcal{L}''$ contains the respective counterparts of all legs in ρ' with the same y, z, k but with an interval that lies behind the last saved τ. If we miss the interval of l', such that $\tau > e_{l'}$, we assume that we cannot push any prior leg any further. In this case, we detect the respective duplications of the legs in ρ' with a higher interval such that the interval of any leg l is greater than $o_{l''}$, where l'' depicts the leg assigned to ρ.

Moreover, we store all checked legs to the vector ρ_{short}. We know that the last leg is incompatible with the prior ones, and can therefore add the following constraint:

$$\sum_{l\in\rho_{short}} x_l + \sum_{l\in\mathcal{L}'} x_l + \sum_{l\in\mathcal{L}''} x_l \leq |\rho_{short}| - 1 \tag{4.40}$$

4.4.3 Bi-objective frameworks

We embed our models into two bi-objective frameworks. For `m1`, `m2`, and `m3` we use the ϵ-constraint method. The branch-and-cut algorithm to solve `m4b` is embedded into both frameworks, namely the ϵ-constraint method and a weighting binary search method.

The ϵ-constraint method

The ϵ-constraint method iteratively solves single-objective problems where one objective is kept in the objective function and the other one is moved to the set of constraints. After each iteration the respective constraint in the constraint set is reduced by a certain ϵ. As we only consider integer variables and coefficients we define the ϵ-value to be 1. For example, let us consider the cost function (4.1) as the main objective, and the preferences objective (4.2) is moved to the constraint as: $\sum_{l\in L} \theta_l x_l \leq \Omega - \epsilon$. Ω is iteratively adapted, inserting the preference value from the previous subproblem and is initially set to ∞.

We solve the problems in a lexicographic order, meaning that in each iteration two MIPs are solved. The algorithm stops once the second extreme point of the Pareto frontier (with the minimal second objective) is reached.

A weighting binary search method

As a second framework we use a binary search in the objective space that is based on the algorithm introduced by Riera-Ledesma and Salazar-González [126]. The idea is to use a weighting method and iteratively enumerate the Pareto frontier. To start the algorithm the extreme points of the Pareto frontier are calculated and stored as $(f_1^{(1)}, f_2^{(1)})$ and $(f_1^{(2)}, f_2^{(2)})$. $f_1^{(1)}$ and $f_1^{(2)}$ give the first, e.g. cost, solutions, of the respective extreme points and conversely $f_2^{(1)}$ and $f_2^{(2)}$ give the value of the second objectives, in our case: preferences. Thus, the objective value is set as $\omega f_1^* + (1-\omega) f_2^*$, where f^* denotes the cost and preference function of the new solution. The weight ω is calculated as $\frac{\alpha}{\alpha+1}$, where $\alpha = \frac{f_2^2 - f_2^1}{f_1^1 - f_1^2}$. At each iteration we add three constraints: (1) $f_1^* < f_1^2$, (2) $f_2^* < f_2^1$, and (3) $\omega f_1^* + (1-\omega) f_2^* \leq \omega f_1^{(1)} + (1-\omega) f_2^{(2)} - 1$. The latter one makes sure that only non-dominated points are generated. The values of the new solutions are then used for the following iterations where the next weights will be calculated with the values $(f_1^{(1)}, f_2^{(1)})$ and (f_1^*, f_2^*), as well as with (f_1^*, f_2^*) and $(f_1^{(2)}, f_2^{(2)})$. The algorithm terminates once no more values can be taken to calculate new weights.

Enhancements For both methods we seize the bi-objective characteristics of our problem: we store the cuts generated in the prior iterations and add them as constraints to the next models. Considering the ϵ-constraint method, we do this within one iteration (the min cost problem receives the cuts from the min preference model), as well as from one iteration to another. As for the binary search, we only solve one MIP with the respective objective function within each iteration. Therefore, we only pass on cuts from one solution to another.

4.5 Computational study

The models and the branch-and-cut algorithm are implemented in C++ and solved with CPLEX 12.9. Tests are carried out using one core of an Intel Xeon Processor E5-2670 v2 machine with 2.50 GHz running Linux CentOS 6.5. Unless otherwise stated a time limit of 12 hours is used.

4.5.1 Test instances

For our computational study we use realistic benchmark instances based on available demographic, spatial and economic data of Vienna, Austria. They are based on those used in Enzi et al. [61] and Enzi et al. [62]. Note that the instances represent a company within a city, thus the data does not aim to replicate the population of the whole city.

One instance set represents a distinct company consisting of one or more offices (or depots) D and users, i.e. employees, P. The number of tasks and their location are randomly generated.

In the original instances, each user may use a subset of the available MOTs $K^p \subseteq K$. Based on this binary assignment of MOTs to users, we generate preference scores on a scale from 1-10, where 1 is best and 10 is worst. For example, if a user has cars in her set of MOTs but no public transportation, then this user will get a lower (better) score on cars and a higher (worse) one on public transportation. The detailed assignments used for the following study is included in Table C10 in the Appendix A.3. In the time-dependent setting, we consider seven different time periods t: pre-rush-hour, rush-hour, after rush-hour, normal day-time traffic, pre-rush-hour, rush-hour and after rush-hour. Here we deduct/add for each preference score a certain number (see Table C11 in the Appendix A.3). Furthermore, we implement an increase/decrease in cost and time for the respective time periods (see Table C11 in the Appendix A.3). For this we assume a factor β which is then multiplied with the base cost. For example, we assume that taking the car during rush-hour takes longer than at noon. We assume $\beta = 1.4$, which is then multiplied with the base cost, e.g. 5. This gives us cost of 7 for the rush-hour for the respective leg. Naturally also the driving times of the legs are adapted accordingly. We calculate a weighted average of cost and preferences if a leg covers more than one periods.

Three different MOTs are considered: car, public transportation including walk, and bike. For our study we assume that all MOTs have an unrestricted capacity. Note that the original setting assumes a limited and fixed pool of cars, which is reasonable for the discussed problem. However, for our first results for the BiO-MMCP we decided to let the number of cars be unlimited, to explore the computational efficiency without restricting the number of shared cars. Distances, time and cost are calculated between all nodes for all modes of transport. Emissions are translated into costs and, together with variable distance cost and cost of time, included into the overall cost calculations and summarized in c_l. The respective preference value θ_l is taken from the above presented values.

Instances are named as `E_`$|P|$`_`I, where $|P|$ is the number of users, and the instance number I is between 0 and 9. For example, the first instance in the set of instances with 20 users ($|P| = 20$) is denoted `E_20_0`. For instance group with $|P|$ users, we solve a set of 10 instances (`E_u_0` to `E_u_9`) and report the average values.

4.5.2 Enhancements and preprocessing

In the following paragraphs we shortly list the enhancements and preprocessing that we conducted.

Relative MIP-gap In our first tests, CPLEX provided weakly dominated solutions or skipped some of the solutions from the Pareto set due to the default relative MIP-gap. Therefore we put a strict MIP-gap tolerance of 0.0000. We compared the output with different tolerances regarding computational efficiency and could not notice a remarkable difference. Therefore, unless otherwise stated, the computational results are based on a MIP-gap tolerance of 0.0000.

Warm start For models `m3` and `m4`, we provide CPLEX with a starting solution. The starting solution is constructed by simply reading the sequence of the tasks as given in

Table 4.2: Average number of nodes ($|\overline{V'}|$), trips ($|\overline{R}|$) and legs ($|\overline{L}|$) for models `m1`, `m2`, `m3`, `m4`, `m4b` and an increasing number of users $|P| = 20, 50, 100, 150, 200, 250, 300$. Row '`m4`,`m4b`+GR' gives the average number of legs after the graph reduction.

	$\|P\| =$	20	50	100	150	200	250	300
$\|\overline{V'}\|$		95	242	476	714	951	1,179	1,422
$\|\overline{R}\|$		31	76	147	218	287	358	427
$\|\overline{L}\|$	`m1`, `m2`	416	1,188	2,612	4,340	6,315	8,734	11,038
	`m3`	947	4,190	13,394	27,756	46,503	70,492	99,462
	`m4`,`m4b`	13,479	41,077	93,079	150,877	216,989	276,626	356,713
	`m4`,`m4b`+GR	3,984	13,995	34,780	61,310	92,790	127,155	167,645

the instance file. For model `m4`, we also track the according times and make sure that the times and intervals match. In the starting solution, public transportation is used on all trips. Moreover, after each iteration we store the solution and provide CPLEX with a MIP start. The MIP start will be infeasible but values can be stored for a possible repair.

Graph reduction Initially for model `m4`, we duplicate each leg every α minutes. Assuming that we have a planning horizon of 12 hours and discretize time in steps of 15 minutes, we end up with 48 duplicates of one leg. However, these legs are very similar to each other or even equal as the time periods t may cover various hours. Therefore, in order to reduce the size of the graph, we merge legs with equal weights.

Table 4.2 gives an overview of the size of the graphs. The table gives information on the introduced models for an increasing number of users ($|P| = 20, 50, 100, 150, 200, 250, 300$). For `m1` and `m2` the underlying graph has the same size as only the preference and cost values on the legs are changing. Row '$|\overline{V'}|$' gives the average number of nodes, '$|\overline{R}|$' the average number of trips, and row '$|\overline{L}|$' the average number of legs in the respective graphs. We observe that the underlying graphs of the first two models have a moderate number of legs as the sequences are predetermined. In models `m3`, `m4`, and `m4b` the sequence is subject of determination which leads to an increasing number of connecting legs, which is increased even further when time-dependency is considered in models `m4` and `m4b`. Row '`m4`,`m4b`+GR' shows the number of the legs in the graph after the graph reduction.

4.5.3 Algorithmic tests

In this section we study the computational efficiency of the introduced variants of the model. We start by comparing the four models (`m1`, `m2`, `m3`, `m4`) in their basic form, i.e., without adding valid inequalities or using the branch-and-cut algorithm. Afterwards we analyze the impact of valid inequalities on model `m3`. Finally, we focus on solving the most challenging model `m4`. We compare the reformulation of `m4` to `m4b`, i.e., if the branch-and-cut algorithm comes with any improvements in computational efficiency. We aim to detect the enhancements by adding valid inequalities, using the branch-and-cut

Table 4.3: Average computational times in seconds for models m1, m2, m3, m3-VI, m4, m4bVIBnCBiOand an increasing number of users $|P| = 20, 50, 100, 150, 200, 250, 300$ for both directions (cost, pref) in the ϵ-constraint method. m3-VI gives results for the respective model with valid inequalities. m4bVIBnCBiOshows results for the model m4b solved by branch-and-cut and passing cuts to subsequent iterations.

	$\|P\| =$	20	50	100	150	200	250	300
m1	cost	0.3	3.0	15.8	45.2	88.1	163.4	247.3
	pref	0.3	3.0	15.8	43.6	85.1	160.9	238.5
m2	cost	0.7	5.5	35.4	92.0	188.2	319.8	428.5
	pref	0.5	4.4	29.8	79.8	165.3	288.2	399.9
m3	cost	7.4	498.5	17,707.6	-	-	-	-
	pref	7.7	1,037.3	-	-	-	-	-
m3-VI	cost	8.0	480.4	5,743.2	-	-	-	-
	pref	8.5	874.5	5,577.5	-	-	-	-
m4	cost	-	-	-	-	-	-	-
	pref	-	-	-	-	-	-	-
m4bVIBnCBiO	cost	3,511.9	-	-	-	-	-	-
	pref	2,730.7	-	-	-	-	-	-

algorithm and choose the best framework (out of the two introduced) to solve model m4b.

Comparison of models m1, m2, m3, and m4

In a first step, we compare the four models regarding their run times. Table 4.3 shows the average computational time in seconds needed to solve an instance group. We first look into results without any valid inequalities or cut generation, given in the rows m1, m2, m3, and m4. The models are embedded in the ϵ-constraint method and enumerated by setting either the cost function as the objective (cost) or the user preferences as the objective (pref). Results are given for instance sets for which we were able to solve all 10 instances. Run times for m1 and m2 are very short. We can solve real-world sized instances with 300 users in less than 5 minutes. For m1 the direction of the ϵ-constraint method has no impact. In the case of m2 setting pref as first objective results in shorter run times. Models m3 and m4 are 'harder' to solve. For model m3 we can see a significant increase in the average run times for the instance group with $|P| = 50$. The largest instance set we can solve comprises 100 users in the case of m3. Adding valid inequalities reduces computational times by a factor of 3 for this instance size (m3-VI) and $|P| = 100$. With m4 we cannot solve any complete instance set. We will go into more detail on m4, its possible extensions and the respective results later. Using the best setting of the proposed branch-and-cut based algorithm, we are able to enumerate the whole Pareto frontier within about 3,000 seconds, on average.

Table 4.4 summarizes the average number of Pareto optimal solutions per instance set. The number of solutions is moderately increasing with number of users $|P|$. Comparing

Table 4.4: Average number of Pareto optimal solutions for models m1, m2, m3, m4 for an increasing number of users $|P| = 20, 50, 100, 150, 200, 250, 300$.

$\|P\| =$	20	50	100	150	200	250	300
m1	18	73	159	250	349	464	518
m2	34	164	346	555	767	993	1,073
m3	18	72	158	-	-	-	-
m4	141	-	-	-	-	-	-

m1 and m3 we see almost the same number of Pareto optimal solutions on average per instance set. If we compare the increased cost in computational complexity coming with m3, we could argue that dissolving the sequences where no time-dependent information is given, is not worthwile. We will investigate the shape of the Parento frontiers in a subsequent section in order to obtain a better understaning of the resulting solutions. Comparing m1 with m2 we can see a distinct increase of optimal solutions on the frontier, even though we only introduced time-dependent cost and preferences. Finally, m4 gives by far the highest number of optimal solutions for the small instance set of $|P| = 20$.

Introducing valid inequalities for model m3

We now analyze the impact of the proposed valid inequalities (VI) in more detail. Table 4.5 presents the computational times in seconds solving m3 without additional information (m3) and by adding valid inequalities (4.21) - (4.27) as well as subtour elimination constraints (4.20) as user cuts (m3-VI). We use both, costs (cost) and preferences (pref) respectively as the 'main' objective function in the ϵ-constraint method. Results are given for $|P| = 100, 150$ and listed for each instance. Row '# solved' shows the number of instances solved with the respective model. We can observe that for some of the instances, e.g. E_100_8, for both cost and pref, the execution time is higher with the valid inequalities than without them. However, on average adding additional information in the form of valid inequalities improves computation times by a factor of approximately 4. Even for instance E_100_2, where we were not able to enumerate the whole Pareto frontier within 12 hours with the base model, we are now able to get the frontiers from either side in less than 3 hours. For the case where $|P| = 150$ and pref we are able to solve all but two instances, however all with relatively long computational times. Direction cost shows longer run times for all solved instances, whereas for two more cases, in total four, the total Pareto frontier cannot be enumerated. None of the instances with $|P| = 150$ has been solved without the valid inequalities. Furthermore, we were not able to solve any of the instances with $|P| = 200$ using m3 or m3-VI.

Solving model m4

We now compare the different approaches for solving m4. Table 4.6 shows the run times for (i) model m4 (m4(ϵ)), (ii) m4 with valid inequalities (m4-VI(ϵ)), (iii) model m4b with valid inequalities, and infeasible path constraints in the form of (4.39)-(4.40) added through cut generation and embedded into the ϵ-constraint method (m4bVIBnC(ϵ)), (iv) the bi-objective branch-and-cut, which is similar to the previous variant but we pass

Table 4.5: Average computational times in seconds for $|P| = 100, 150$ solving `m3` without valid inequalities (`m3`) and the same model including valid inequalities (`m3`-VI) and having either cost (`cost`) or preferences (`pref`) set as the objective function in the ϵ-constraint method. '# solved' shows the number of instances solved. TO = time out.

	`cost`		`pref`			`cost`		`pref`	
	`m3`	`m3`-VI	`m3`	`m3`-VI		`m3`	`m3`-VI	`m3`	`m3`-VI
E_100_0	**2,419**	2,852	3,212	2,628	E_150_0	TO	TO	TO	TO
E_100_1	3,329	3,104	**2,513**	2,777	E_150_1	TO	TO	TO	**39,507**
E_100_2	35,758	**7,275**	TO	8,504	E_150_2	TO	TO	TO	**41,675**
E_100_3	3,382	3,544	**3,281**	3,892	E_150_3	TO	37,060	TO	**20,507**
E_100_4	24,162	**4,177**	25,159	4,553	E_150_4	TO	TO	TO	TO
E_100_5	27,186	6,655	25,587	**6,062**	E_150_5	TO	19,766	TO	**14,098**
E_100_6	11,152	9,814	11,403	**9,092**	E_150_6	TO	18,352	TO	**13,640**
E_100_7	24,112	9,245	22,085	**6,880**	E_150_7	TO	38,193	TO	**21,112**
E_100_8	3,398	5,204	**2,685**	3,678	E_150_8	TO	30,083	TO	**23,643**
E_100_9	22,771	**5,561**	28,090	7,708	E_150_9	TO	TO	TO	**35,617**
# solved	10	10	9	10		0	5	0	8

the cuts generated as constraints from one solution to another (`m4bVIBnCBiO`(ϵ)), (v) model `m4b` solved by branch-and-cut embedded in the weighting binary search method (`m4bVIBnC`(ω)), and (vi) the branch-and-cut used to solve `m4b` using the weighting binary search method and passing cuts to subsequent iterations (`m4bVIBnCBiO`(ω)). Again all results are given for both directions, `cost` and `pref` in the case of the ϵ-constraint scheme. In the case of the binary search, both objectives are combined in one weighting objective function. Times are in seconds. Row '# solved' gives the number of instances solved. Results are given for each instance for $|P| = 20$.

Using model `m4`(ϵ) and the direction `cost`, only one instance is solved, using `pref` as the main objective, two instances can be solved within 12 hours of computation time. Adding valid inequalities (`m4-VI`(ϵ)), we are able to increase the number of instances solved to 6 for the direction `cost` and to 7 for the direction `pref`. Still for most of the instances the run times exceed 10,000 seconds.

Moving from the model with the time variables (`m4`) to the entirely integer model (`m4b`) with cut generation, we can improve run times considerably by at least a factor of 10 (column `m4bVIBnC`(ϵ)). Yet, we are still not able to enumerate the whole frontier for instance E_20_9. By seizing the bi-objective character of the model and handing over detected infeasible paths as constraints from one iteration of the ϵ-constraint scheme to the next, we further increase in the algorithms' computational efficiency (`m4bVIBnCBiO`(ϵ)). Note that different to most works, we add the detected infeasible paths not to a cut pool but explicitly to the set of constraints. All instances with $|P| = 20$ can now be solved for `m4`. The last two columns of Table 4.6 show the results obtained by applying the weighting method and conducting a binary search in the objective space. It is again clearly visible, that the approach where cuts are passed on from iteration to another, enhances computation times and thus seizing the bi-objective character of the models is beneficial. Nevertheless, the run times are not comparable to '`m4bVIBnCBiO`(ϵ)'. The

Table 4.6: Average computational times in seconds for each instance of $|P| = 20$ solving m4 using different approaches. The first four sets of results (m4(ϵ), m4-VI(ϵ), m4bVIBnC(ϵ), m4bVIBnCBiO(ϵ)) are solved using the ϵ-constraint method with either cost (cost) or preferences (pref) set as the objective function. m4(ϵ) gives the results by solving m4 without any additional information, m4-VI(ϵ) adds valid inequalities to the model, m4bVIBnC(ϵ) solves the integer model by branch-and-cut, and m4bVIBnCBiO(ϵ) passes detected infeasible paths as constraints to next iteration. Columns m4bVIBnC(ω) and m4bVIBnCBiO(ω) show the results received by the weighting binary search. The latter one again passes former detected infeasible paths as constraints to next subproblems. '# solved' shows the number of instances solved. TO = time out.

	m4(ϵ)		m4-VI(ϵ)		m4bVIBnC(ϵ)		m4bVIBnCBiO(ϵ)		m4bVIBnC(ω)	m4bVIBnCBiO(ω)
	cost	pref	cost	pref	cost	pref	cost	pref		
E_20_0	TO	26,780	12,663	7,190	779	743	206	**192**	669	324
E_20_1	TO	TO	TO	TO	2,957	2,099	455	**407**	1,823	621
E_20_2	TO	20,437	13,504	9,339	737	793	168	**159**	608	192
E_20_3	TO	TO	29,772	18,012	856	890	192	**189**	711	294
E_20_4	TO	TO	TO	23,490	2,447	1,798	473	**420**	1,681	560
E_20_5	TO	TO	19,823	10,810	743	781	327	**190**	712	322
E_20_6	TO	TO	31,309	26,751	1,982	1,557	399	**369**	1,244	590
E_20_7	TO	TO	TO	TO	3,383	2,436	628	**496**	2,555	716
E_20_8	TO	TO	32,790	18,522	892	948	333	**256**	729	414
E_20_9	TO	TO	TO	TO	TO	TO	31,937	**24,629**	TO	TO
# solved	0	2	6	7	9	9	10	10	9	9

reason is that the binary search algorithm calls the solver approximately twice as often as the ϵ-constraint.

As noted, instance E_20_9 requires significantly more time for computing the Pareto frontier than all the others. The reason is that it is the only instance with $|P| = 20$ which has one user with three trips. The total number of trips or average number of trips per person are in line with the other instances. Thus, the maximum number of trips per user has a significant impact.

Note that we add all found infeasible paths to the set of constraints instead of adding them to a cut pool. As the number of cuts generated is relatively small, and is also decreasing over time, the additional constraints are of a manageable size. However, we have tried both approaches and computational times confirmed the efficiency of our approach.

The above results show that 'm4bVIBnCBiO(ϵ)' (with direction pref) is, for our problem setting, more efficient than 'm4bVIBnCBiO(ω)'. As discussed, this is mainly due to the increase in the number of MIPs that have to be solved. Table 4.7 compares the run times of the two approaches for $|P| = 50$. The table shows similar results as above. The ϵ-constraint method is able to solve more instances and also, if the instance is solved by both approaches, results in shorter computation times.

As we have seen, it is beneficial to exploit the bi-objective nature of the underlying optimization problem by using previously generated cuts in subsequent iterations. In Figure 4.2, we show the number of cuts added at each iteration for one chosen instance, namely E_20_0. Figures 4.2(a) and (b) show the results for the ϵ-constraint method, first without adding the cuts as constraints at each iteration and then by using the generated cuts in the respective submodels. Figures 4.2(c) and (d) give the number of cuts added

Table 4.7: Average computational times for each instance for $|P| = 50$ solving `m4b` by branch-and-cut embedded in the ϵ-constraint method (`m4bVIBnCBiO`(ϵ)) and in the weighting binary search algorithm (`m4bVIBnCBiO`(ω)). Both approaches add prior detected infeasible paths as constraints to model.

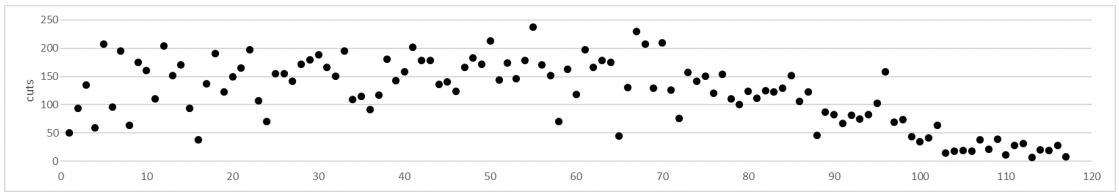
	`m4bVIBnCBiO`(ϵ)	`m4bVIBnCBiO`(ω)
E_50_0	16,712	31,100
E_50_1	TO	TO
E_50_2	TO	TO
E_50_3	13,759	22,657
E_50_4	5,876	13,954
E_50_5	41,764	TO
E_50_6	TO	TO
E_50_7	TO	TO
E_50_8	4,746	7,907
E_50_9	TO	TO

Table 4.8: Average number of cuts added at each iteration for instances with $|P| = 20$ solving `m4b` by branch-and-cut embedded in the ϵ-constraint method (`m4bVIBnC`(ϵ)) or the weighting binary search (`m4bVIBnC`(ω)), and by adding detected infeasible paths constraints to the model (`m4bVIBnCBiO`(ϵ), `m4bVIBnCBiO`(ω))

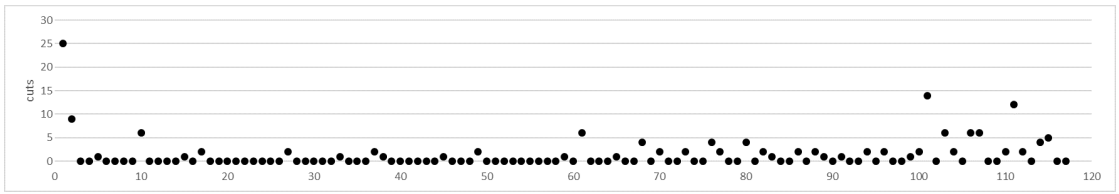
$\lvert P\rvert = 20$	E_20_0	E_20_1	E_20_2	E_20_3	E_20_4	E_20_5	E_20_6	E_20_7	E_20_8	E_20_9
`m4bVIBnC`(ϵ)	120.0	205.1	180.2	223.0	227.5	179.1	208.3	189.8	179.8	1,694.9
`m4bVIBnCBiO`(ϵ)	1.3	3.3	2.1	5.2	7.6	6.8	7.9	4.3	3.2	70.2
`m4bVIBnC`(ω)	49.8	105.0	68.6	94.1	109.7	67.3	89.6	97.7	63.0	-
`m4bVIBnCBiO`(ω)	1.4	2.4	1.2	1.5	4.0	2.9	4.5	2.2	1.6	-

at each iteration for the weighting method conducting a binary search. As we can see, solving each subproblem individually generates a much higher number of cuts at each iteration, whereas in the other case, where we propagate cuts from iteration to iteration, we drastically reduce the cuts added at each subproblem. This is valid for both methods. Moreover, by comparing Figures 4.2(b) and (d), we see that the binary search method actually produces fewer cuts in later iterations. The reason is that the binary search method detects solutions, where infeasibility needs to be proven. This also results in two times more iterations for this method. Nevertheless, we can clearly observe that for either approach, the additional information from prior iterations has a remarkable impact on cut generation iterations.

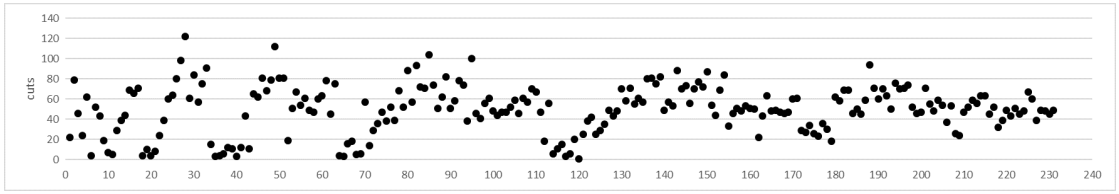
Table 4.8 gives the number of cuts added per iteration on average for both the ϵ-constraint method as well as the binary search approach for each instance in the set with $|P| = 20$. We show the case where each iteration is using only the current information (`m4bVIBnC`(ϵ), `m4bVIBnC`(ω)) and where we use information in the form of cuts added as constraints from prior iterations (`m4bVIBnCBiO`(ϵ), `m4bVIBnCBiO`(ω)). We can clearly see that without additional information we use up to 100 times more cuts. As discussed prior, the binary search method has a lower average number, but more iterations are conducted.



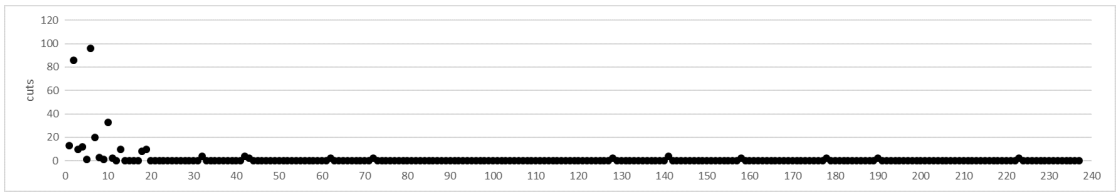
(a) m4bVIBnC(ϵ)



(b) m4bVIBnCBiO(ϵ)



(c) m4bVIBnC(ω)



(d) m4bVIBnCBiO(ω)

Figure 4.2: Number of cuts added at each iteration for instance E_20_0. m4bVIBnC(ϵ) solves model m4b by branch-and-cut embedded in the ϵ-constraint method, and m4bVIBnCBiO(ϵ) additionally stores the detected infeasible paths to the set of constraints. m4bVIBnC(ω) solves model m4b by branch-and-cut and the weighting binary search method, and m4bVIBnCBiO(ω) additionally passes infeasible path constraints to subsequent iterations.

4.5.4 Managerial insights

We briefly discuss managerial implications. We start by looking at the respective Pareto frontiers for a chosen set of instances. Then we continue by studying the different MOT compositions when solving the different variants of the model. Note that the models provide the decision maker with a range of trade-off solutions. Based on this solution pool, the decision maker derives actions and takes the best solution fitting to their requirements.

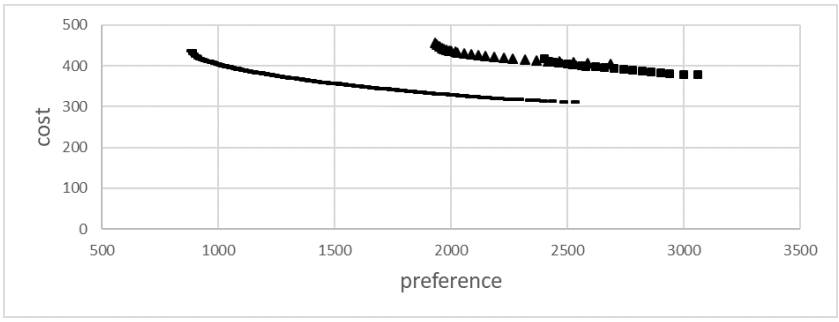
Comparison of Pareto frontiers for models `m1`, `m2`, `m3`, and `m4`

Figure 4.4 shows the Pareto frontiers for instances E_20_0 and E_20_9. The x-axis represents preferences, y-axis cost. Note that for both instances only three frontiers are visible. This is because the frontier of `m1` is hidden behind `m3`. For these small instances, the additional freedom to choose sequences of tasks and trips is not giving any improvement to the model. Frontiers for `m2` are similar in their shape for both instances, however slightly differ in their relation to the other curves, especially to `m3`(`m1`). Introducing time-dependent values for `m2`, lower (better) overall preferences but higher cost are obtained, visible as a shift to the left on the x-axis and a shift upwards on the y-axis. The increased cost come from the additional time needed during specific day-times. Note that we usually have a $\beta > 1$, meaning that we rarely decrease the driving time compared to the base scenario (except for public transportation, where we assume shorter cycle times for, e.g., rush-hours). For `m4`, the length of the frontier exceeds all the other curves. It is clearly visible, that with time-dependent preferences and cost as well as flexible sequences, we have a greater set of Pareto optimal solutions. Also, the curve is shifting to the left corner, meaning that we have better overall cost as well as preferences. The average cost and preference values for instances with $u = 20$ are: 505 and 2,878 for `m1`, 548 and 2,272 for `m2`, 505 and 2,878 for `m3`, 476 and 1,591 for `m4`, respectively. Concluding, we can say that time-dependencies do have a great impact when solving the bi-objective multimodal car-sharing problem. Furthermore, we observe that only dissolving the fixed sequence does not come with high improvements but in combination with time-dependencies a greater amount of solutions as well as lower cost and better user satisfaction is obtained.

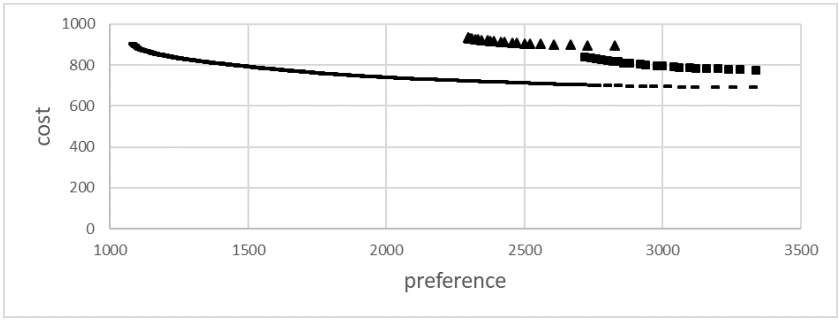
MOT assignment for models `m1`, `m2`, `m3`, and `m4`

Finally, let us have a closer look at the MOTs assigned. We analyze the number of trips covered by each MOT (car, bike, public transportation), for two instances, namely E_20_0 and E_20_9, for all four models `m1`, `m2`, `m3`, `m4`. In Figure 4.5 we show the respective Pareto frontier for the four models, and include the number of trips taken by each MOT for the respective Pareto optimal solution. Note that the number of trips that are covered by a car does not have to be equal to the number of cars used in total as a car might take more than one trip during a day.

Starting with `m4`, we observe a similar development for both instances for all MOTs. With increasing (worse) preferences, and decreasing cost, we gradually assign more cars and less bikes. The number of trips taken by public transportation is more or less constant. Thus, most cost-efficient, considering time-dependencies, are car trips, best



(a) E_20_0



(b) E_20_9

(c) `m4bVIBnC`(ϵ)

Figure 4.4: Pareto frontiers for models `m1`, `m2`, `m3`, `m4` solving instances E_20_0 and E_20_9. The y-axis represents cost, preferences are on the x-axis.

preferences give bike trips. A car is, in our instance set, the fastest mode of transport. As we include time in the cost function, this also makes cars often the cheapest option. Moreover, our study gives relatively good time-dependent preference scores to bikes, as it is, e.g., good in rush-hours to avoid congestion or overcrowded public transportation. This of course, has a great impact on the resulting tendencies in the final results.

For the other models, the picture is slightly different and instance-dependent. Generally, we can say for `m1`, `m2`, and `m3` the number of trips taken by bike is decreasing with lower cost and higher (worse) preferences. The number of trips taken by public transportation are increasing with higher (worse) preferences and lower cost.

Comparing the extreme points of all Pareto frontiers for all models regarding their composition we can conclude: for `m1` and `m3` we always assign more cars and public transportation to the cost optimal solution (except for one instance for `m3`), the number of trips taken by public transportation and cars decreases with higher cost but better preferences. Bikes are preferred by the preference optimal solutions, and increase with less cost. Also for `m2` we can observe that the number of bikes assigned is decreasing with increasing cost and lower (better) preferences. The opposite holds for public transportation. We can figure an unchanged level of trips being assigned to public transportation for `m4`. For `m4` lower cost and higher (worse) preferences lead to more cars assigned and, conversely, more bikes are assigned with an increase in cost, and decrease in preferences.

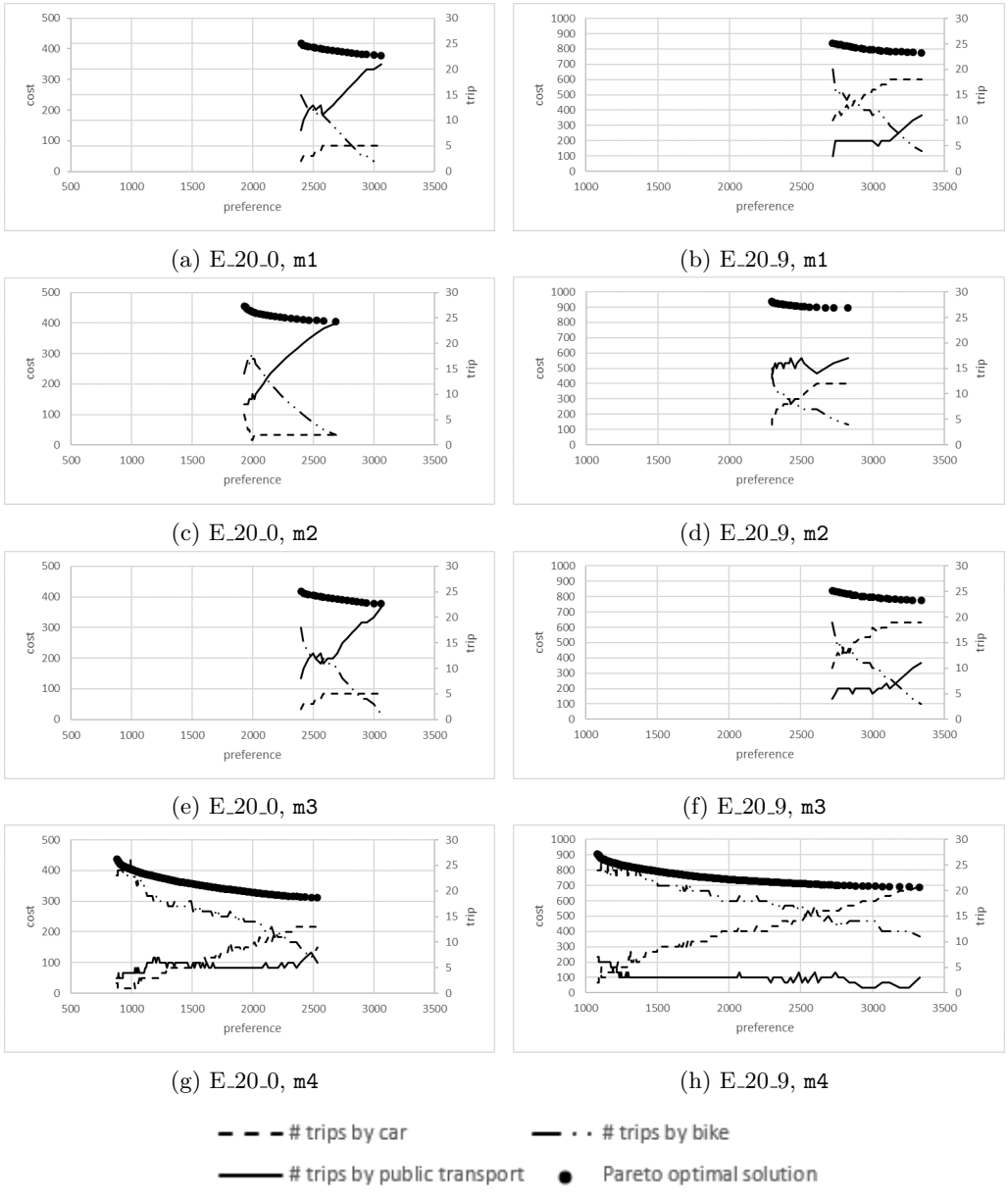
Table 4.9 provides a better overview of the MOTs assigned to trips for each instance set and model. The numbers are given as averages over all instances within an instance set. Rows 'av' provide the average of the average number of trips by the respective MOT (car, bike, public). 'min' gives the average of the minimum number of trips conducted by the respective MOT, and 'max' gives the average maximum number. The results are organised by model (`m1`, `m2`, `m3`, `m4`), MOTs, and number of users $|P| = 20, 50, 100, 150, 200, 250, 300$.

Generally, we observe that bikes are very often assigned and used for the highest number of trips on average. `m3` assigns about the same amount of cars and public transportation. `m2` always shows the highest number of trips taken by public transportation. Thus, by having the choice between MOTs for a trip with a fixed sequence, public transportation is preferred. `m4` has a very high number of trips taken by bikes.

Note that the composition of the mobility offers varies a lot among the models. Furthermore, the difference between the minimums and maximums of the assigned MOTs is usually very high, which means that the solutions are changing considerably over the course of the Pareto frontier. This means that, from a decision makers perspective, considering the proposed trade-offs and variants of the problem has a big impact on the MOTs used in a mobility system. Assigning different MOTs influences the user-centred objective to a great extent. With this results we can confirm the relevance of this study and conclude that it is highly beneficial to consider not only cost but also user-preferences when operating a shared mobility system.

4.6 Conclusion and future work

Inspired by the change in mobility patterns we study the bi-objective multimodal car-sharing problem where we assign modes of transport to trips as well as cars and user



(a) E_20_0, `m1`

(b) E_20_9, `m1`

(c) E_20_0, `m2`

(d) E_20_9, `m2`

(e) E_20_0, `m3`

(f) E_20_9, `m3`

(g) E_20_0, `m4`

(h) E_20_9, `m4`

(i) `m4bVIBnC`(ϵ)

Figure 4.5: Number of trips assigned for each Pareto optimal solution by the respective mode of transport (car, bike, public transportation) for models `m1`, `m2`, `m3`, `m4` solving instances E_20_0 and E_20_9.

Table 4.9: Average values of average number of MOT assigned to trips (av), minimum (min) or maximum (max) for models `m1`, `m2`, `m3`, `m4` for an increasing number of users $|P| = 20, 50, 100, 150, 200, 250, 300$. Considered modes of transport are car, bike and public transportation.

		`m1`			`m2`			`m3`			`m4`		
	$\|P\|$	car	bike	public	car	bike	public	car	bike	public	car	bike	public
av		7.7	11.7	11.4	2.2	11.6	17.1	8.1	10.5	12.1	6.3	20.0	4.5
min	20	4.5	4.7	6.8	0.8	5.0	11.5	4.7	3.9	7.5	1.8	12.1	2.5
max		10.6	19.3	15.9	4.5	16.8	23.6	10.9	18.3	16.4	14.6	26.3	6.8
av		23.4	29.9	23.0	3.1	27.9	45.3	23.3	27.6	25.3	21.6	45.0	10.0
min	50	11.0	11.5	14.4	1.8	11.2	29.1	11.8	9.1	16.4	5.3	29.8	5.1
max		33.9	50.6	32.3	6.5	43.6	61.1	34.9	46.2	34.9	41.0	62.0	14.5
av		47.1	54.2	45.3	5.9	54.2	86.5	47.3	54.0	45.4	-	-	-
min	100	22.5	20.8	30.1	2.9	19.9	54.1	23.8	19.9	31.0	-	-	-
max		71.7	93.9	57.0	13.2	82.8	120.0	72.0	87.9	61.4	-	-	-
av		72.7	82.7	62.2	10.8	81.4	125.4	66.4	86.9	64.8	-	-	-
min	150	37.5	34.3	41.8	6.4	32.7	82.3	33.5	35.5	45.6	-	-	-
max		106.1	138.2	79.4	21.0	121.0	171.4	102.4	136.5	86.4	-	-	-
av		94.6	108.6	83.9	15.4	107.4	164.3	-	-	-	-	-	-
min	200	48.5	39.5	55.1	10.3	40.1	105.4	-	-	-	-	-	-
max		142.1	183.3	107.0	25.3	163.1	227.3	-	-	-	-	-	-
av		120.0	137.0	101.2	20.9	137.1	200.3	-	-	-	-	-	-
min	250	62.2	53.0	65.3	14.3	51.5	129.5	-	-	-	-	-	-
max		177.7	230.4	130.4	37.7	201.8	279.6	-	-	-	-	-	-
av		131.2	163.8	132.5	19.6	163.2	244.7	-	-	-	-	-	-
min	300	70.3	62.7	88.4	13.8	57.8	158.5	-	-	-	-	-	-
max		194.7	268.7	171.9	36.6	239.7	347.3	-	-	-	-	-	-

routes. As objectives we consider costs and user-centred preferences. Both objectives are, depending on the variant of the model, studied with time dependencies. We model different cost/times as well as preferences during a day, as people might want to avoid driving through rush-hour by car. We introduce four different variants of the model where we gradually dissolve a fixed sequence of tasks and trips as well as introduce the effect of the time-dependent values. The increase in flexibility in the model comes with an increase in the complexity as well as a an increase in the number of Pareto optimal solutions. Therefore, we reformulate the last variant, without fixed sequences and time-dependencies, to a purely integer model and propose a branch-and-cut algorithm. We show that our branch-and-cut algorithm can enumerate the Pareto frontier for prior non-tractable instances within seconds. We embed the algorithm into two bi-objective frameworks, namely the ϵ-constraint method and a weighting binary search method. We show that adding previously detected infeasible path constraints to subsequent iterations reduces computational times considerably. In our computational study we observe that only dissolving the fixed sequence does not come with high improvements. However, in combination with time-dependencies a greater amount of solutions as well as lower cost and better user satisfaction is obtained. Moreover, we observe that the solutions change significantly along the Pareto frontier. This confirms the relevance of this study. We conclude that it is highly beneficial to consider not only cost but also user-preferences when operating a shared mobility system.

Even though we are able to show a significant enhancement in computational efficiency for a set of instances, our approach has limitations. Enumerating the whole Pareto frontier for instances with users having more than two trips throughout a day, seems challenging. Thus, future work should tackle this issue by focusing on the development of a separation algorithm adjusted to these specific characteristics. Moreover, specific matheuristics where the relative MIP-gap is increased or the ϵ value adapted, may lead to promising further improvement in run times. Furthermore, the development of metaheuristics should enable an increase in computational efficiency for the proposed problem. Finally, as this work only optimizes average scores of preferences, a min-max approach is planned for future work in order to improve the integration of preferences on a user level.

Acknowledgements

This work has been partially funded by the Climate and Energy Funds (KliEn) under grant number 853767. This research was funded in whole, or in part, by the Austrian Science Fund (FWF) [P 31366]. For the purpose of open access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

Chapter 5

Conclusion

Inspired by the change in mobility, we studied shared mobility systems within a closed group. We introduced efficient modeling approaches and solution techniques for vehicle sharing and extended the idea by letting user co-ride with each other as well as introducing the "human factor" into our optimization problem. We formulated the problems using and extending well-known models, and applied state-of-the-art algorithms to solve real-world sized instances. In our problem setting we assumed a company, and a set of users covering meetings/tasks during the day. The user-to-task assignment was always fixed, depending on the problem we also determined start times of the tasks. We considered that the company has a fixed pool of cars available that can be shared, and other modes of transport can be used too. Even though the initial framework of our study depicted a corporate environment, it can easily be applied to any specific network with a predefined set of users in a closed community, and is therefore of high importance in current and future concepts of shared mobility systems. In the following we shortly summarize the three parts of this thesis and outline some further extensions and research directions.

In Chapter 2 we introduced the vehicle-sharing problem. We assumed fixed times of the tasks, no interchangeable trips of users, and trips being covered by a certain mode of transport. We assumed that either one vehicle type or multiple ones are shared. The objective was to assign the shared vehicles to the trips such that savings by using, e.g., a car instead of any other mobility type, are maximized. We assumed that if a trip was not covered by a shared vehicle, the user would take the cheapest other MOT for this trip. We formulated the case where only one type of vehicle is shared as the maximization equivalent of the minimum-cost flow problem, and the case where multiple types of vehicle may be shared as a maximum multi-commodity flow problem. Even though these problems are proven to be NP-hard, state-of-the-art commercial solvers are able to solve real-world sized instances. We provided a thorough analysis discussing managerial implications. We gave insights into the different kinds of shared vehicles, and the number of trips per car during a day; we analyzed the effect of sharing and no sharing, included user preferences, and compared different objective function. To sum up, we can say that, if possible, electric vehicles are preferred, a car takes 1.5 to 2.8 trips per day, sharing is only beneficial if a big enough car pool is available, and it is crucial to have operational cost and cost of time combined in the objective function.

In Chapter 3 we proposed the multimodal car- and ride-sharing problem where a company is providing car/vehicle-sharing but also users may co-ride with each other. We again considered fixed timing of the tasks, but now users would exclude MOTs from their set of MOTs. We aimed to assign a pool of shared cars and joined rides, such that savings were maximized when using a car instead of any other MOT. We again only assigned cars and assumed that the trips not covered by a car, would be taken by the cheapest other MOT calculated for this trip. We formulated the problem as an extended

vehicle scheduling problem and introduced a two-stage decomposition approach. In the first stage we enumerated all possible trips (including all ride-sharing possibility), whereas in the second stage we found car routes through an efficient column generation based approach. A car route may consist of several trips throughout the day. With our approach we were able to circumvent the complexity of modeling ride-sharing and solved real-world sized instances in reasonable time, making it possible to use it on a daily basis. Nevertheless, the algorithm and the problem statement have its restriction. To start with, we only considered one type of shared cars. This can be extended to multiple ones. Furthermore, we considered fixed sequences and timings of tasks. To ensure more flexibility and even greater gain, this can be dissolved and further savings studied. Furthermore, future work might want to have changes of drivers outside of the depots. Even though this restriction was given as inconvenient by project partners, it can be worth a try to see if further enhancements are possible. Lastly, the stochasticiy might be a crucial extension of the problem, as the tasks of drivers or co-riding persons might be delayed, which then ruins the plan of trips.

In Chapter 4 we studied the bi-objective multimodal car-sharing problem. We introduced user preferences in a second objective where the users would score the MOTs on an individual level. The chapter introduced four different variants of the problem. Depending on the variant, we also assumed time-dependent travel times as well as user preferences, and assumed an increased level of flexibility of timings, i.e., not fixed times of the tasks. In this chapter we not only assigned the shared cars/vehicles to the trips but assigned all MOTs to trips and scheduled car and user routes. The increased flexibility and time-dependent values came with an increase in the complexity as well as the number of Pareto optimal solutions. For the most complex model we proposed a branch-and-cut algorithm and embedded it in two bi-objective frameworks, namely the ϵ-constraint method and a weighting binary search method. We showed that by handing over generated cuts as hard constraints to subsequent iterations, we can achieve an improvement in computational efficiency. In the computational study we saw that the solutions change significantly along the Pareto frontier. Nevertheless, we also detected some restrictions in the chapter as solving instances with more than two trips per user was challenging. Future work might want to focus on more advanced separation algorithms to circumvent this problem. Moreover, we found a lot of Pareto optimal solutions for the case where we had no fixed timing of tasks and assumed time-dependent preferences and travel times / cost. Thus, developing heuristics to enumerate parts of the frontier might help to decrease the computational times.

Chapter A

Appendix

A.1 Modeling and solving a vehicle-sharing problem

Table A1: Parameter value setting for the instances. The total cost are calculated as ((sloping factor * cost per km) + (sloped distance * (1 / average speed) + setup time) * cost per time + (cost of emissions * emissions per km)).

sloping factor:	foot: 1.1 bike: 1.3 car: 1.3 public transportation: 1.5
CO2 emissions per km in gramm:	foot: 0 bike: 0 combustion engine car: 200.9 electric car: 42.7 public transportation: 0
cost of CO2 emissions:	5 euro/t
average speed (km/h):	foot: 5 bike: 16 car: 30 public transportation: 20
cost per km:	foot: 0 bike: 0 combustion engine car: 0.188 electric car: 0.094 public transportation: 0 taxi: 1.2
cost per time:	19.42 euro per hour
setup time (in minutes):	foot: 0 bike: 2 car: 10 public transportation: 5 taxi: 5

Table A2: Comparison of total cost split into combustion engine cars (car-type1) and other MOTs, and savings for increasing number of u and m for *VShP-1T:car*. Share of total cost of the respective car and MOT costs given in 'car-type1 / total' and 'other MOTs / total'.

u	m	car-type1	other MOTs	total	savings	car-type1 / total	other MOTs / total
20	4	155	401	555	- 49	0.28	0.72
	8	243	294	537	- 68	0.45	0.55
	20	316	204	520	- 85	0.61	0.39
	40	318	202	520	- 85	0.61	0.39
50	4	264	1,423	1,686	- 122	0.16	0.84
	8	467	1,160	1,626	- 182	0.29	0.71
	20	846	704	1,550	- 258	0.55	0.45
	40	1,050	482	1,532	- 276	0.69	0.31
100	4	291	3,090	3,381	- 178	0.09	0.91
	8	553	2,735	3,288	- 270	0.17	0.83
	20	1,155	2,008	3,163	- 395	0.37	0.63
	40	1,820	1,289	3,109	- 449	0.59	0.41
150	4	320	5,146	5,466	- 221	0.06	0.94
	8	575	4,752	5,327	- 360	0.11	0.89
	20	1,235	3,832	5,067	- 619	0.24	0.76
	40	2,079	2,797	4,875	- 811	0.43	0.57
200	4	338	7,087	7,424	- 241	0.05	0.95
	8	624	6,635	7,260	- 406	0.09	0.91
	20	1,320	5,608	6,928	- 738	0.19	0.81
	40	2,310	4,339	6,648	- 1,017	0.35	0.65
250	4	373	9,063	9,436	- 289	0.04	0.96
	8	670	8,567	9,238	- 487	0.07	0.93
	20	1,470	7,391	8,861	- 864	0.17	0.83
	40	2,506	6,048	8,555	- 1,170	0.29	0.71
300	4	350	9,966	10,316	- 247	0.03	0.97
	8	638	9,486	10,124	- 439	0.06	0.94
	20	1,427	8,348	9,775	- 789	0.15	0.85
	40	2,495	7,017	9,512	- 1,051	0.26	0.74

Table A3: Comparison of total cost split into electric cars (car-type2) and other MOTs, and savings for increasing number of u and m for *VShP-1T:ecar*. Share of total cost of the respective car and MOT costs given in 'car-type2 / total' and 'other MOTs / total'.

u	m	car-type2	other MOTs	total	savings	car-type2 / total	other MOTs / total
20	4	162	379	541	- 64	0.30	0.70
	8	271	243	514	- 91	0.53	0.47
	20	348	143	491	- 114	0.71	0.29
	40	352	138	491	- 114	0.72	0.28
50	4	243	1,420	1,663	- 145	0.15	0.85
	8	430	1,155	1,588	- 220	0.27	0.73
	20	850	625	1,475	- 333	0.58	0.42
	40	1,098	340	1,438	- 370	0.76	0.24
100	4	266	3,089	3,355	- 204	0.08	0.92
	8	509	2,731	3,239	- 319	0.16	0.84
	20	1,084	1,977	3,062	- 497	0.35	0.65
	40	1,776	1,175	2,951	- 608	0.60	0.40
150	4	298	5,139	5,437	- 250	0.05	0.95
	8	539	4,736	5,275	- 412	0.10	0.90
	20	1,150	3,808	4,957	- 729	0.23	0.77
	40	1,960	2,733	4,693	- 993	0.42	0.58
200	4	310	7,083	7,393	- 272	0.04	0.96
	8	577	6,625	7,202	- 463	0.08	0.92
	20	1,237	5,572	6,809	- 857	0.18	0.82
	40	2,165	4,279	6,444	- 1,222	0.34	0.66
250	4	351	9,050	9,401	- 324	0.04	0.96
	8	625	8,551	9,176	- 549	0.07	0.93
	20	1,371	7,357	8,728	- 997	0.16	0.84
	40	2,338	5,994	8,332	- 1,393	0.28	0.72
300	4	322	9,962	10,284	- 280	0.03	0.97
	8	596	9,470	10,066	- 497	0.06	0.94
	20	1,332	8,315	9,646	- 917	0.14	0.86
	40	2,339	6,952	9,291	- 1,272	0.25	0.75

Table A4: Comparison of total cost split into combustion engine cars (car-type1), electric cars (car-type2) and other MOTs, and savings for increasing number of u and m for *VShP-xT*. Share of total cost of the respective car and MOT costs given in 'car-type / total' and 'other MOTs / total'.

$u =$	m	car-type1	car-type2	other MOTs	total	savings	car-type1 /total	car-type2 /total	other MOTs /total
20	4	64	91	392	547	- 58	0.12	0.17	0.72
	8	88	163	272	522	- 82	0.17	0.31	0.52
	20	48	298	149	495	- 110	0.10	0.60	0.30
	40	2	347	142	491	- 114	0.00	0.71	0.29
50	4	112	139	1,421	1,673	- 135	0.07	0.08	0.85
	8	194	249	1,159	1,603	- 205	0.12	0.16	0.72
	20	270	536	694	1,500	- 308	0.18	0.36	0.46
	40	196	859	401	1,456	- 352	0.13	0.59	0.28
100	4	124	152	3,090	3,366	- 193	0.04	0.05	0.92
	8	235	291	2,734	3,260	- 299	0.07	0.09	0.84
	20	468	637	1,998	3,102	- 456	0.15	0.21	0.64
	40	634	1,102	1,270	3,006	- 552	0.21	0.37	0.42
150	4	142	161	5,146	5,450	- 237	0.03	0.03	0.94
	8	234	314	4,748	5,296	- 391	0.04	0.06	0.90
	20	490	691	3,820	5,000	- 686	0.10	0.14	0.76
	40	800	1,191	2,771	4,762	- 925	0.17	0.25	0.58
200	4	155	168	7,085	7,408	- 258	0.02	0.02	0.96
	8	267	331	6,628	7,226	- 439	0.04	0.05	0.92
	20	530	740	5,586	6,856	- 810	0.08	0.11	0.81
	40	902	1,314	4,306	6,522	- 1,144	0.14	0.20	0.66
250	4	161	193	9,063	9,417	- 308	0.02	0.02	0.96
	8	281	363	8,558	9,202	- 523	0.03	0.04	0.93
	20	587	818	7,376	8,781	- 944	0.07	0.09	0.84
	40	956	1,442	6,018	8,416	- 1,309	0.11	0.17	0.72
300	4	154	179	9,964	10,298	- 265	0.01	0.02	0.97
	8	271	337	9,482	10,091	- 473	0.03	0.03	0.94
	20	570	790	8,338	9,698	- 866	0.06	0.08	0.86
	40	969	1,424	6,983	9,376	- 1,187	0.10	0.15	0.74

Table A5: Solution time in seconds for *VShP-1T:car*, *VShP-1T:ecar*, *VShP-xT* for an increasing number of u and m.

u	m	*VShP-1T:ecar*	*VShP-1T:car*	*VShP-xT*
20	4	0.0	0.0	0.1
	8	0.0	0.0	0.1
	20	0.0	0.0	0.0
	40	0.0	0.0	0.1
50	4	0.1	0.2	0.4
	8	0.1	0.2	0.4
	20	0.2	0.2	0.4
	40	0.1	0.2	0.4
100	4	0.7	0.7	1.6
	8	0.7	0.7	1.6
	20	0.7	0.7	1.6
	40	0.7	0.7	1.6
150	4	1.6	1.6	3.7
	8	1.6	1.6	3.6
	20	1.6	1.6	3.7
	40	1.6	1.6	3.6
200	4	3.0	3.1	7.0
	8	3.1	3.1	6.8
	20	3.1	3.1	6.7
	40	3.0	3.0	6.9
250	4	5.0	4.9	11.0
	8	4.8	5.0	10.8
	20	4.9	4.9	11.0
	40	4.8	5.0	10.9
300	4	7.3	7.4	16.3
	8	7.2	7.6	17.1
	20	7.3	7.4	16.8
	40	7.4	7.4	16.8

Table A6: Average cost for car-type1 (=combustion engine cars) and other MOTs, in total, and average savings for *VShP-1T:car* and the different preference variants (prefVar0-prefVar6). The values are given for an increasing number of u and averaged over $m = 4, 8, 20, 40$.

u =	20	50	100	150	200	250	300
				VShP-1T:car			
car-type1	258	656	955	1,052	1,148	1,255	1,227
other MOTs	275	942	2,281	4,132	5,917	7,767	8,704
total	533	1,599	3,235	5,184	7,065	9,022	9,932
savings	- 72	- 209	- 323	- 503	- 601	- 703	- 631
				prefVar0			
car-type1	180	429	624	783	825	902	872
other MOTs	424	1,495	3,361	5,930	8,302	10,854	12,093
total	604	1,925	3,985	6,713	9,127	11,757	12,964
savings	- 215	- 595	- 1,119	- 1,682	- 1,939	- 2,109	- 2,264
				prefVar1			
car-type1	237	580	873	989	1,080	1,162	1,169
other MOTs	323	1,084	2,475	4,358	6,198	8,154	9,014
total	560	1,664	3,348	5,347	7,279	9,316	10,184
savings	- 76	- 209	- 343	- 590	- 698	- 818	- 787
				prefVar2			
car-type1	42	200	359	510	618	726	722
other MOTs	558	1,570	3,150	5,047	6,893	8,783	9,669
total	600	1,770	3,509	5,558	7,511	9,509	10,391
savings	- 11	- 56	- 74	- 196	- 213	- 310	- 252
				prefVar3			
car-type1	144	425	707	848	943	1,036	1,030
other MOTs	437	1,281	2,698	4,569	6,415	8,331	9,231
total	581	1,706	3,405	5,417	7,358	9,367	10,261
savings	- 39	- 149	- 245	- 444	- 471	- 642	- 522
				prefVar4			
car-type1	245	605	901	1,003	1,101	1,187	1,188
other MOTs	325	1,084	2,496	4,475	6,327	8,375	9,220
total	571	1,689	3,397	5,478	7,429	9,562	10,408
savings	- 84	- 251	- 409	- 713	- 811	- 981	- 910
				prefVar5			
car-type1	42	209	376	522	650	754	747
other MOTs	559	1,565	3,142	5,053	6,882	8,788	9,682
total	601	1,774	3,518	5,575	7,532	9,542	10,429
savings	- 11	- 63	- 84	- 233	- 255	- 376	- 306
				prefVar6			
car-type1	148	442	732	868	963	1,059	1,045
other MOTs	439	1,278	2,700	4,622	6,474	8,445	9,337
total	587	1,720	3,432	5,489	7,437	9,504	10,382
savings	- 45	- 170	- 278	- 533	- 551	- 755	- 615

Table A7: Average cost for one car-type1 (combustion engine cars), car-type2 and other MOTs, in total and average savings for *VShP-xT*and the different preference variants (prefVar0-prefVar6). The values are given for an increasing number of u and averages over all m.

u =	20	50	100	150	200	250	300
				VShP-xT			
car-type1	50	193	365	417	463	496	491
car-type2	225	446	545	589	638	704	683
other MOTs	239	919	2273	4121	5901	7754	8692
total	514	1558	3184	5127	7003	8954	9866
savings	-91	-250	-375	-560	-663	-771	-698
				preVar0			
car-type1	27	86	163	303	304	343	339
car-type2	147	295	387	455	477	518	505
other MOTs	418	1520	3401	5916	8303	10848	12075
total	593	1900	3951	6673	9084	11709	12919
savings	-226	-620	-1153	-1722	-1982	-2156	-2309
				preVar1			
car-type1	110	253	413	471	510	534	558
car-type2	130	330	440	486	535	588	573
other MOTs	313	1058	2461	4354	6193	8150	9010
total	553	1641	3314	5311	7238	9272	10141
savings	-83	-232	-377	-626	-738	-862	-829
				prefVar2			
car-type1	31	96	172	254	289	336	348
car-type2	13	114	191	246	324	376	371
other MOTs	555	1553	3133	5040	6874	8768	9646
total	599	1763	3496	5539	7488	9481	10365
savings	-12	-63	-88	-214	-236	-338	-278
				prefVar3			
car-type1	64	180	330	398	451	483	491
car-type2	80	257	374	432	466	524	505
other MOTs	432	1252	2675	4554	6404	8323	9227
total	576	1689	3379	5384	7321	9329	10222
savings	-44	-165	-271	-477	-508	-680	-560
				prefVar4			
car-type1	139	332	454	516	543	596	593
car-type2	106	270	447	487	558	592	595
other MOTs	325	1087	2496	4475	6327	8374	9220
total	571	1689	3397	5478	7429	9562	10408
savings	-84	-251	-409	-713	-811	-981	-910
				prefVar5			
car-type1	41	139	260	323	363	399	383
car-type2	1	75	116	194	277	348	361
other MOTs	559	1560	3142	5057	6892	8795	9685
total	601	1774	3518	5575	7532	9542	10429
savings	-11	-63	-84	-233	-255	-376	-306
				prefVar6			
car-type1	88	253	382	439	482	527	515
car-type2	59	192	353	428	481	532	531
other MOTs	439	1274	2698	4623	6473	8445	9337
total	587	1720	3432	5489	7437	9504	10382
savings	-45	-170	-278	-533	-551	-755	-615

Table A8: Total cost comparison for OF:time split into combustion engine cars (car-type1) and other MOTs for an increasing number of u and m for *VShP-1T:car*.

	u =	20	50	100	150	200	250	300
4	car-type1	186	275	293	323	341	374	355
	other MOTs	383	1,455	3,176	5,473	7,479	9,691	10,380
	total	569	1,730	3,469	5,796	7,820	10,065	10,735
8	car-type1	310	482	560	593	640	690	665
	other MOTs	235	1,166	2,774	4,980	6,922	9,047	9,751
	total	545	1,648	3,334	5,573	7,562	9,738	10,416
20	car-type1	418	954	1,192	1,279	1,370	1,482	1,482
	other MOTs	108	602	1,973	3,891	5,702	7,656	8,405
	total	526	1,556	3,165	5,169	7,071	9,138	9,887
40	car-type1	418	1,277	1,971	2,161	2,389	2,585	2,605
	other MOTs	107	268	1,145	2,738	4,298	6,127	6,947
	total	526	1,545	3,116	4,899	6,687	8,712	9,552

Table A9: Comparison of total cost for OF:time split into car-type1, car-type2 (= combustion engine and electric cars) and other MOTs for an increasing number of u and m for *VShP-xT*.

	u =	20	50	100	150	200	250	300
4	car-type1	98	140	144	157	174	195	184
	car-type2	79	123	135	151	152	166	156
	other MOTs	385	1,455	3,176	5,473	7,479	9,689	10,379
	total	562	1,718	3,455	5,781	7,805	10,049	10,719
8	car-type1	158	249	274	284	318	357	321
	car-type2	138	212	262	283	293	301	309
	other MOTs	236	1,167	2,773	4,978	6,922	9,049	9,755
	total	532	1,628	3,309	5,546	7,533	9,708	10,385
20	car-type1	204	482	605	643	694	745	743
	car-type2	190	433	537	581	620	673	676
	other MOTs	114	602	1,973	3,890	5,698	7,655	8,403
	total	508	1,517	3,115	5,114	7,012	9,073	9,822
40	car-type1	248	689	1,003	1,072	1,224	1,305	1,296
	car-type2	156	537	886	995	1,066	1,170	1,197
	other MOTs	107	271	1,146	2,740	4,296	6,125	6,946
	total	511	1,497	3,035	4,806	6,587	8,600	9,439

A.2 Modeling and solving the multimodal car- and ride-sharing problem

In the following we outline a sample solution in order to give the reader a better idea of the proposed model and the resulting changes. We show the results for instance E_20_0, $m = 4$ and two depots. We consider 20 users and 88 nodes (start/end nodes of a trip and tasks), thus 4.4 nodes on average per user. The instance comprises 28 trips, giving 1.4 trips per user. Figure A.1 shows the instance with its trips. As we see, many trips go from the depot, to a meeting and back again. Assuming no ride-sharing but only car-sharing, each trip will be covered either by car or the cheapest other MOT. However, as we can see, there is a high potential of merging trips. As the task-to-user assignment is fixed, the best option is to allow for ride-sharing.

Note that we have short and/or many ride-sharing activities on a route. This gives unclear graphs. For the route of car 4, we give a more detailed explanation with the figures below. This should give a good idea of the benefits of this model. But let us start by shortly sketching car routes 1, 2 and 3 in words. The driver of car 1 is user 4, covering both of her trips as a driver. Users 1, 3, 16, and 20 are co-riding for parts of their trips. Car 2 is driven by user 7 and handed over to user 6 for a second trip. Users 9 and 13 are sharing the ride with user 7, user 15 is twice co-riding with user 6. The driver of car 3 is user 10 for two trips. The first trip is conducted without a person joining the ride, on the second trip users 18 and 19 co-ride in the car.

Figure A.2 provides one out of the four car routes given in the solution of instance E_20_0. Figure A.2(b) shows the route of car 4 after solving the MMCRP, Figure A.2(a) shows the prior individual trips of the users. There are in total five trips of users 5, 10, 11 and 19. Note that user 5 has two trips in this example. The number next to the nodes in Figure A.2(b) gives the sequence of the route. User 5 is the driver, the other users are co-riding. The route starts in a depot (stop 1), brings user 11 to the meeting (stop 2), and continues to the meeting of the driver (user 5, stop 3). After the meeting, the driver picks up user 19 (stop 4) and takes the user to the next meeting (stop 5), before returning to the depot (stop 6). After a break at the depot, user 5 drives to her last meeting (stop 7) and returns to the depot again (stop 8). Note that the respective users get/leave from their meetings with the cheapest other MOT, not included in this visualization. Figure A.3 gives the solution in a time-space network.

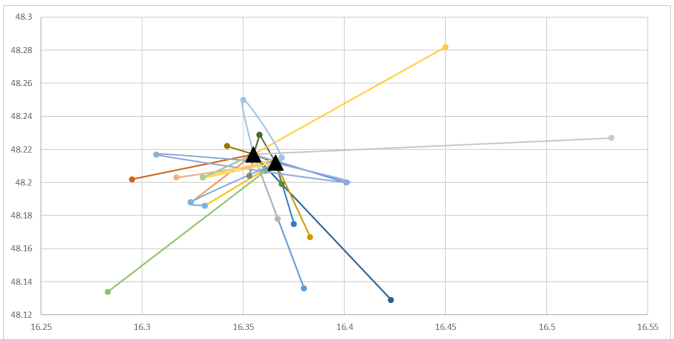
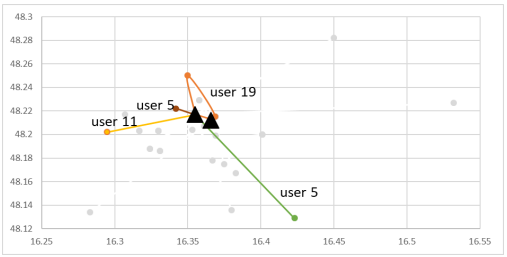
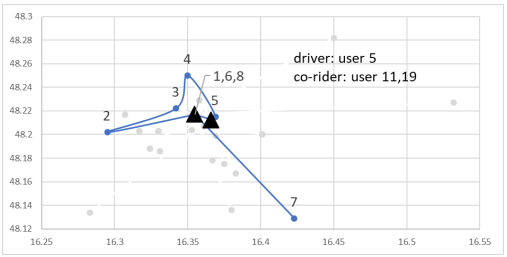


Figure A.1: Instance E_20_0 showing the trips of the users without considering ride-sharing.



(a) Trips of users that are (partially) merged to one car route after solving the MMCRP.



(b) Route of car 4 after solving the MMCRP.

Figure A.2: Partial solution of instance E_20_0 before and after solving the MMCRP.

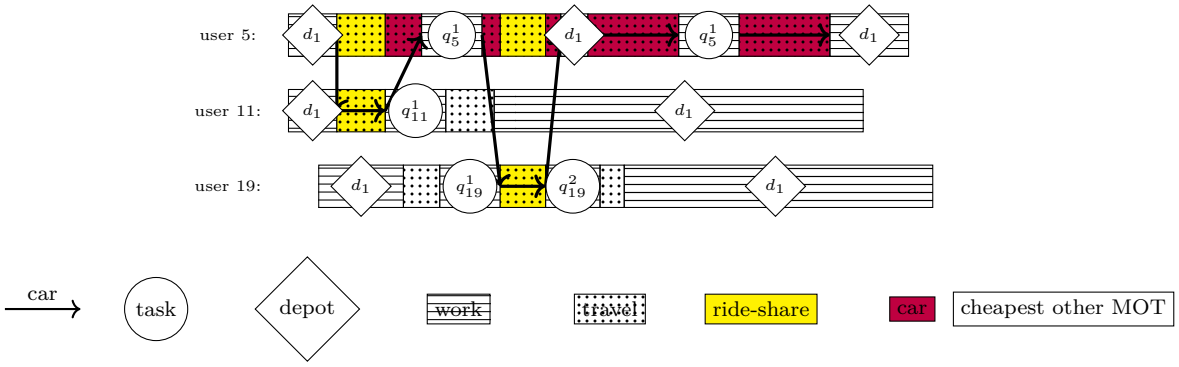
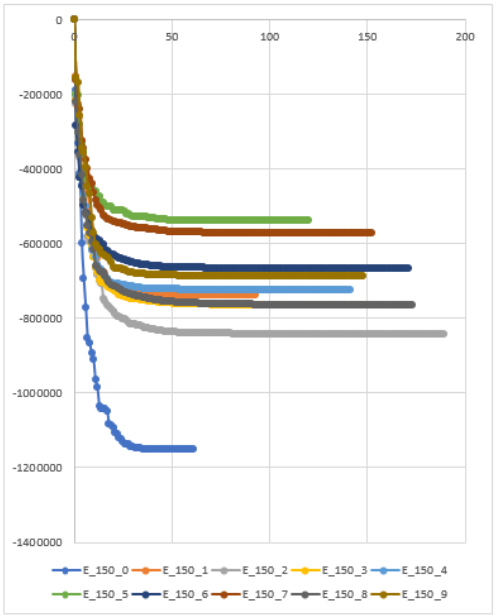
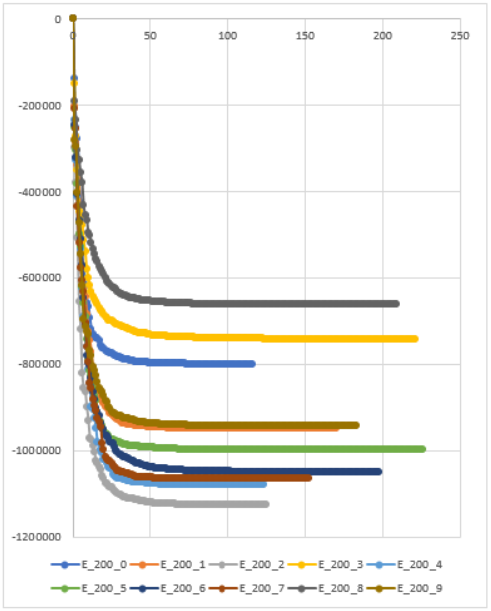


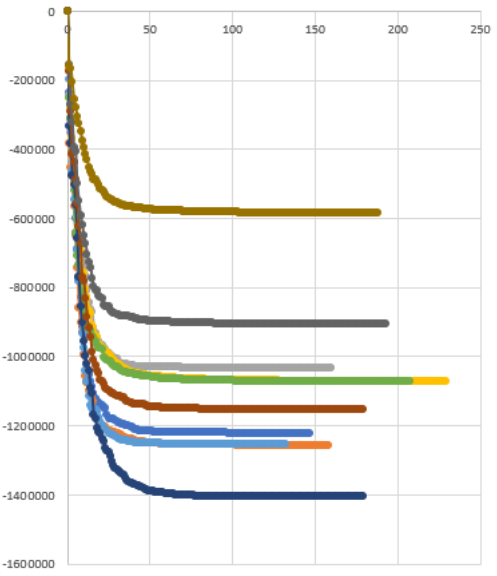
Figure A.3: Solution of instance E_20_0 after solving the MMCRP represented in a time-space network. Depots are denoted in a diamond shape (d_1), tasks are represented as circles and denoted as q_p^i. Background rectangles with lines depict duration of a meeting, dots mean the user is traveling. If the background is not colored, the user is traveling with the cheapest other MOT, purple depicts travel by car, yellow ride-sharing. The arrows show the routes of the car.



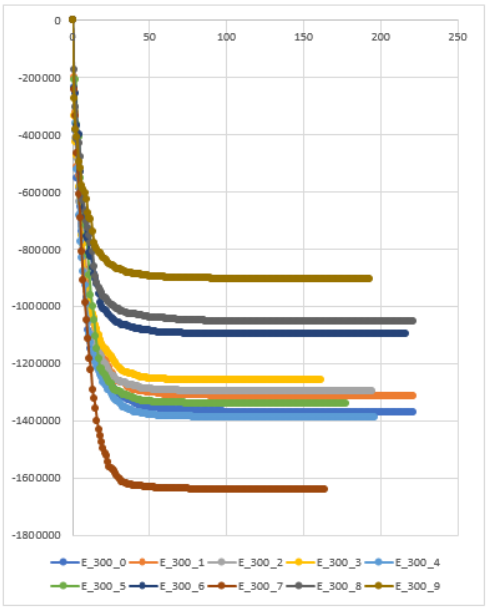
(a) $u = 150$



(b) $u = 200$

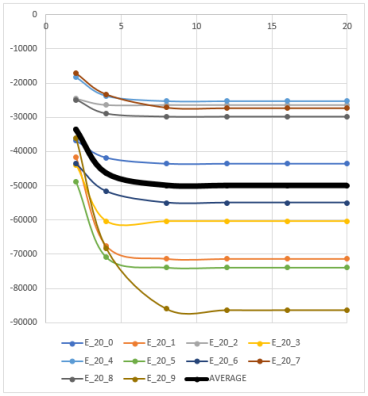


(c) $u = 250$

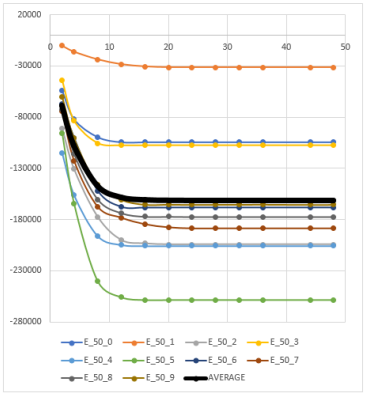


(d) $u = 300$

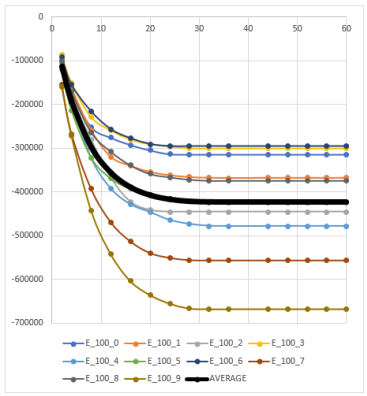
Figure A.4: Convergence of the column generation algorithm for increasing number of users (u). The y-axis shows the computational time in seconds, the x-axis represents the number of iterations.



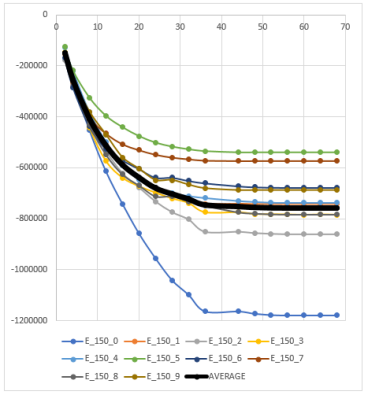
(a) $u = 20$



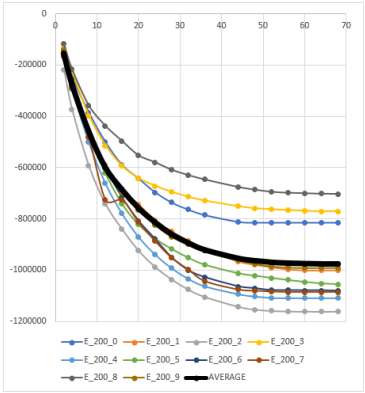
(b) $u = 50$



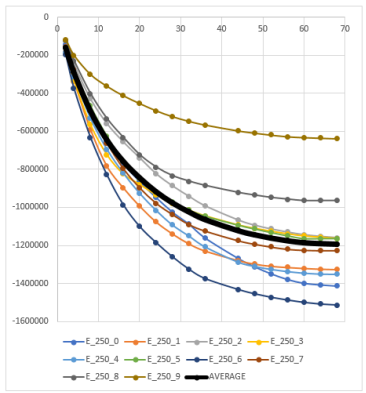
(c) $u = 100$



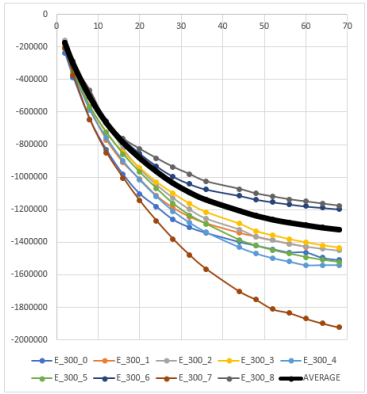
(d) $u = 150$



(e) $u = 200$



(f) $u = 250$



(g) $u = 300$

Figure A.5: Optimal fleet size for increasing number of users u. The x-axis shows the number of cars used, the y-axis represents the obtained savings.

A.3 The bi-objective multimodal car-sharing problem

forall $p \in P$ **do**
  construct empty route vector ρ;
  $v = \gamma_p$;
  while $v \neq \phi_p$ **do**
    forall $l \in$ v.outgoing **do**
      if(l.person $= p$ & $\overline{x}_l = 1$) store leg l in vector ρ;
      $v = l$.endNode;
      break;
    end
  end
  $\Delta = \infty$; $\tau = 0$; $\mathcal{W} = 0$; F = 0, $l = 1$;
  forall $l \leq |\rho_l|$ **do**
    if (l.startNode = trip start node a) $\Delta = \infty$; $F = 0$; $\mathcal{W} = 0$;
    $\tau = \tau + s_v + t_{l-1}$;
    $\mathcal{W}$ += min{max{0,e_l-τ},h};
    Δ = min{Δ, max{0,e_l-(τ+$\mathcal{W}$)}};
    F = $\mathcal{W}$ + Δ;
    if $o_l \leq \tau \leq e_l$ **then**
      l++;
    else
      $\tau' = \tau$;
      τ += min{max{0,$o_l - \tau$}, F};
      if $o_l \leq \tau \leq e_l$ **then**
        Δ += min{$\mathcal{W}$ - ($\tau - \tau'$),0};
        $\mathcal{W}$ = max{$\mathcal{W}$ - ($\tau - \tau'$),0};
        l++;
      else
        add Cut (see Section 4.4.2);
      end
    end
  end
end

Algorithm 1: Separation of user routes.

```
forall m ∈ K : m = car do
    trips = 0;
    forall v ∈ D do
        forall l ∈ v.outgoing do
            if (l.mot ≠ m) continue;
            construct empty vector ρ;
            if(x̄_l = 1) store leg l in vector ρ;
            v = l.endNode;
            while v ≠ carEndNode do
                forall l ∈ v.outgoing do
                    if (l.mot != m) continue;
                    if(x̄_l = 1) store leg l in vector ρ;
                    v = l.endNode;
                    if (v = startNode) trips++;
                    break;
                end
            end
            if (trips < 2) continue;
            Δ = ∞; τ = 0; W = 0; F = 0, l = 1;
            forall l ≤ |ρ_l| do
                if (l.startNode = trip start node a) Δ = ∞; F = 0; W = 0;
                τ = τ + s_{l.startNode} + t_{l−1};
                W += min{max{0,e_l-τ},h};
                Δ = min{Δ, max{0,e_l-(τ+W)}};
                F = W + Δ;
                if o_l ≤ τ ≤ e_l then
                    l++;
                else
                    τ' = τ;
                    τ += min{max{0,o_l − τ}, F};
                    if o_l ≤ τ ≤ e_l then
                        Δ += min{W - (τ − τ'),0};
                        W = max{W - (τ − τ'),0};
                        l++;
                    else
                        add Cut (see Section 4.4.2);
                    end
                end
            end
        end
    end
end
```

Algorithm 2: Separation of car routes.

Table C10: Preference scores assigned based on the binary assignment from the instance generation.

binary assignment of MOTs					preference scores		
walk	bike	car	e-car	public	walk/public	bike	car
1	0	0	0	0	4	6	7
0	1	0	0	0	6	4	7
0	0	0	0	1	4	6	7
0	0	1	0	0	6	7	4
0	0	0	1	0	6	7	5
1	1	0	0	0	4	4	7
0	1	0	0	1	4	4	7
0	0	1	0	1	4	5	4
0	0	1	1	0	7	7	4
1	0	0	0	1	4	6	7
0	1	1	0	0	6	4	4
0	0	0	1	1	4	7	5
1	0	1	0	0	4	5	4
0	1	0	1	0	6	5	6
1	1	0	0	1	4	4	7
1	0	0	1	0	4	7	5
0	1	1	0	1	4	4	4
0	0	1	1	1	7	7	4
1	1	1	0	1	4	4	4
0	1	1	1	1	7	4	4
1	0	1	1	1	4	7	4
1	1	1	1	0	4	4	4
1	1	0	1	1	4	4	7
1	1	1	1	1	4	4	4
0	0	0	0	0	4	5	5

Table C11: On the left: Adaption of the user preferences for the time-dependent values for each time period t. The base values are taken from Table C10 and accordingly deducted/added. On the right: β-values to multiply the respective cost and time value of the respective MOT for the respective time periods t.

t	walk/public	bike	car
0	-3	-2	+1
1	+2	-2	+3
2	-2	-1	-3
3	+ 0	+ 0	+ 0
4	-2	-3	+1
5	+2	-2	+3
6	-1	+2	-2

car	walk/public	bike
1.2	1.1	1
1.4	0.8	1.1
1.3	0.9	1
1	1	1
1 .3	1	1
1.4	0.9	1.1
1.1	1.3	1

A.3.1 Heuristic solutions for the BiO-MMCP

In Chapter 4.3 we have discussed the computational limits when solving the BiO-MMCP to optimality. Even though we were able to reduce the computational times applying advanced techniques, there were still some limitations, especially when considering m3-VI and m4bVIBnCBiO and an increased number of users. Therefore, in what follows, we discuss results obtained from solving m3-VI(ϵ) and m4bVIBnCBiO(ϵ) heuristically. Unless otherwise stated, we always take preferences as the main objective, and put the cost function into the set of constraints. We will not use a purely heuristic approach but a matheuristic, to seize the obtained enhancements in the Chapter 4.3. The aim is to decrease computational times without losing too much of the solution quality. In order to do so, we will increase the relative MIP-gap between the upper and lower bound for the respective models, as introduced by [142]. As we solve each subproblem in a lexicographic order, we have two models per iteration. For each of these models a different gap can be applied. In the following the gap of the first model is given as λ_1, the gap of the second one as λ_2.

Table C12 shows the run times for m3-VI(ϵ), $|P| = 100$ and the respective λ values. The first column gives the respective instance name, the second one the computational times for m4bVIBnCBiO(ϵ) with a gap of 0, the other columns give the respective run times with gap λ. The best average run times shows the setting with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$ with 2,001 seconds compared to 5,77 with the original MIP-gap. In order to obtain an idea on the solution quality, we calculate the hypervolume indicator [156]. The hypervolume measures the volume (or if it a bi-objective optimization problem the surface) of the obtained Pareto frontier bounded by a given reference point. With this, frontiers can be compared with each other, and determine the proximity of herustically obtained frontiers to its optimal counterpart.

Table C12: Computational times for m3-VI(ϵ) after increasing the relative MIP gap of the models, where λ_1 gives the gap for the first MIP within an iteration and λ_2 the MIP-gap of the the second one. Results are given for each instance with $|P| = 100$.

	λ_1, λ_2 0.00,0.00	λ_1, λ_2 0.01,0.01	λ_1, λ_2 0.01,0.05	λ_1, λ_2 0.05,0.01	λ_1, λ_2 0.05,0.05	λ_1, λ_2 0.01,0.1	λ_1, λ_2 0.1,0.01
E_100_0	2,628	1,283	1,768	1,262	1,745	2,055	**1,250**
E_100_1	2,777	1,419	2,095	**1,402**	2,094	2,099	1,455
E_100_2	8,504	4,251	4,275	3,452	3,818	4,253	**3,234**
E_100_3	3,892	1,777	2,804	1,410	2,308	2,409	**1,409**
E_100_4	4,553	2,340	2,718	1,665	2,595	2,837	**1,662**
E_100_5	6,062	3,070	3,482	**2,657**	3,295	3,421	2,759
E_100_6	9,092	3,472	3,744	2,678	3,306	3,735	**2,674**
E_100_7	6,880	3,380	3,199	2,862	2,988	3,320	**1,648**
E_100_8	3,678	1,821	2,289	**1,688**	2,209	2,313	1,850
E_100_9	7,708	3,207	3,909	**2,048**	4,016	3,843	2,065
average	5,577	2,602	3,028	2,112	2,837	3,029	**2,001**

Table C13 gives the respective hypervolume per instance (hv) as well as how close we are to the optimal solution, shown as a percentage (%) for all settings with different

λ values. The last row shows the average over the presented instances. We see that most of the solutions are very close to the optimal frontier, mostly showing 99% of the hypervolume compared to the optimal solution. There are two exceptions, namely settings $\lambda_1 = 0.05$,$\lambda_2 = 0.01$ and $\lambda_1 = 0.1$,$\lambda_2 = 0.01$ with less than 98%. As we saw previously, the latter setting is the best regarding run times. Thus, for `m3`-VI the fastest setting comes with a loss of solution quality.

Table C13: Hypervolume (hv) and comparison to the base case with a MIP-gap of 0 (%) solving `m3`-VI for each instance with $|P| = 100$ with the respective MIP-gaps λ.

	λ_1,λ_2 0.00,0.00 hv	λ_1,λ_2 0.01,0.01 %	λ_1,λ_2 0.01,0.05 %	λ_1,λ_2 0.05,0.01 %	λ_1,λ_2 0.05,0.05 %	λ_1,λ_2 0.01,0.1 %	λ_1,λ_2 0.1,0.01 %
E_100_0	346020	0.992	0.992	0.992	0.991	0.992	0.992
E_100_1	372720	0.992	0.992	0.992	0.993	0.992	0.992
E_100_2	963820	0.993	0.992	0.991	0.984	0.992	0.979
E_100_3	430480	0.992	0.992	0.961	0.987	0.992	0.961
E_100_4	448230	0.993	0.993	0.973	0.979	0.993	0.973
E_100_5	560360	0.991	0.990	0.988	0.987	0.991	0.989
E_100_6	626340	0.991	0.991	0.987	0.972	0.991	0.987
E_100_7	923260	0.993	0.993	0.991	0.985	0.993	0.937
E_100_8	442930	0.989	0.989	0.968	0.976	0.988	0.968
E_100_9	559250	0.992	0.992	0.953	0.957	0.992	0.953
average	567341	0.992	0.992	0.979	0.981	0.992	0.973

Table C14 shows the run times obtained with $\lambda_1 = \lambda_2 = 0.01$ for each instance with $|P| = 150, 200$. The given setting can solve almost all but one instance for $|P| = 150$ and one for $|P| = 200$ as well as decrease the computational times. Other λ-settings might give better run times (see above), however would lead to a lower solution quality. Therefore we refrain at this point from trying further settings of λ for `m3`-VI.

Table C14: Computational times for each instance with $|P| = 150, 200$ for `m3`-VI and MIP-gap $\lambda_1 = \lambda_2 = 0.01$.

$\|P\| =$	150	200
_0	TO	TO
_1	17,294	TO
_2	15,833	TO
_3	8,253	28,796
_4	20,813	TO
_5	7,497	TO
_6	8,968	TO
_7	10,734	TO
_8	9,145	TO
_9	17,840	TO

We try the same approach for our model `m4bVIBnCBiO` with the branch-and-cut and adding cuts as constraints at each iteration. Table C15 summarizes the computational times for each instance with $|P| = 20$ and the different λ configurations. Again, the last row shows the averages over all listed instances. We reduce computational times with all settings, whereas we have a low of 691 seconds compared to 2,731 seconds for the base case. By looking at the above discussed most challenging instance E_20_9, we even reduce from over 24,000 seconds to little above 5,000. However, as above, we have a closer look at the respective hypervolumes to evaluate the quality of the Pareto frontiers.

Table C15: Computational times for `m4bVIBnCBiO`(ϵ) after increasing the relative MIP gap of the models, where λ_1 gives the gap for the first MIP within an iteration and λ_2 the MIP-gap of the the second one. Results are given for each instance with $|P| = 100$.

	λ_1, λ_2 0.00, 0.00	λ_1, λ_2 0.01, 0.01	λ_1, λ_2 0.01,0.05	λ_1, λ_2 0.05,0.01	λ_1, λ_2 0.05, 0.05	λ_1, λ_2 0.01,0.1	λ_1, λ_2 0.1,0.01
E_20_0	192	172	176	160	176	176	**158**
E_20_1	407	338	358	247	308	369	**242**
E_20_2	159	153	147	132	133	146	**123**
E_20_3	189	165	153	133	154	152	**130**
E_20_4	420	367	335	260	362	343	**231**
E_20_5	190	161	175	107	155	174	**104**
E_20_6	369	336	366	241	304	362	**230**
E_20_7	496	409	432	314	415	444	**284**
E_20_8	256	222	230	189	230	229	**176**
E_20_9	24,629	17,335	15,802	8,145	12,004	17,490	**5,234**
average	2,731	1,966	1,817	993	1,424	1,988	**691**

In Table C16 the respective hypervolumes and comparison to the optimal solution (%) is given. As previously, we can observe a good approximation for all settings of λ. Even the prior rather poor approximation of $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$, shows better results now. As this is at almost 99% of the optimal solution, we take this settings now for a further analysis of bigger instances.

Table C17 presents the results for $|P| = 50, 100$ with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$ solving `m4bVIBnCBiO`(ϵ). All but one instance with $|P| = 100$ can be solved within the stated time limit. Some of the instances can even be solved in reasonable time of less than one hour. A further increase in the size of the instances ($|P| = 100$) is not efficiently solvable, whereas only one can be solved within 12 hours. However, considering the first run times obtained for `m4bVIBnCBiO` this is already an achievement.

Table C16: Hypervolume (hv) and comparison to the base case with a MIP-gap of 0 (%) solving `m4bVIBnCBiO`(ϵ) for each instance with $|P| = 20$ with the respective MIP-gaps λ.

	λ_1, λ_2	λ_1, λ_2	λ_1, λ_2	λ_1, λ_2	λ_1, λ_2	λ_1, λ_2	λ_1, λ_2
	0.0,0.00	0.01,0.01	0.01,0.05	0.05,0.01	0.05,0.05	0.01,0.1	0.1,0.01
	hv	%	%	%	%	%	%
E_20_0	142545	0.998	0.998	0.995	0.996	0.998	0.995
E_20_1	234728	0.998	0.998	0.991	0.995	0.998	0.990
E_20_2	167284	0.999	0.998	0.995	0.996	0.998	0.991
E_20_3	164209	0.998	0.998	0.993	0.996	0.998	0.991
E_20_4	172932	0.998	0.998	0.988	0.992	0.998	0.982
E_20_5	147538	0.998	0.998	0.990	0.991	0.998	0.983
E_20_6	181043	0.997	0.998	0.989	0.993	0.998	0.987
E_20_7	304919	0.998	0.998	0.994	0.997	0.998	0.988
E_20_8	222476	0.998	0.998	0.993	0.996	0.998	0.992
E_20_9	361842	0.999	0.999	0.990	0.991	0.999	0.971
average		0.998	0.998	0.992	0.994	0.998	0.987

Table C17: Computational times for each instance with $|P| = 150, 200$ for `m4` and MIP gap $\lambda_1 = 0.1, = \lambda_2 = 0.01$.

$\|P\| =$	50	100
_0	4,488	TO
_1	TO	TO
_2	11,149	TO
_3	6,866	TO
_4	1,384	TO
_5	4,718	TO
_6	5,956	40,383
_7	37,265	TO
_8	1,838	TO
_9	4,512	TO

Bibliography

[1] Abad, H. K. E., Vahdani, B., Sharifi, M., and Etebari, F. (2018). A bi-objective model for pickup and delivery pollution-routing problem with integration and consolidation shipments in cross-docking system. Journal of Cleaner Production, 193:784 – 801.

[2] Adelgren, N. and Gupte, A. (2017). Branch-and-bound for biobjective mixed integer programming.

[3] Adler, J. and Mirchandani, P. (2016). The vehicle scheduling problem for fleets with alternative-fuel vehicles. Transportation Science, 51.

[4] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (2001). Minimum cost flow problemminimum cost flow problem. In Floudas, C. A. and Pardalos, P. M., editors, Encyclopedia of Optimization, pages 1382–1392. Springer US, Boston, MA.

[5] Al Maghraoui, O., Vallet, F., Puchinger, J., and Yannou, B. (2019). Modeling traveler experience for designing urban mobility systems. Design Science, 5:e7.

[6] Alexiou, D. and Katsavounis, S. (2015). A multi-objective transportation routing problem. Operational Research, 15(2):199–211.

[7] Anderluh, A., Nolz, P. C., Hemmelmayr, V. C., and Crainic, T. G. (2019). Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone' customers arising in urban logistics. European Journal of Operational Research.

[8] Androutsopoulos, K. N. and Zografos, K. G. (2017). An integrated modelling approach for the bicriterion vehicle routing and scheduling problem with environmental considerations. Transportation Research Part C: Emerging Technologies, 82:180 – 209.

[9] Aneja, Y. P. and Nair, K. P. K. (1979). Bicriteria transportation problem. Management Science, 25(1):73–78.

[10] APA-OTS (2020). Grüne/hammer: Autofreie innenstadt ist wichtiger erfolg für die menschen in wien. `https://www.ots.at/presseaussendung/OTS_20200617_OTS0086/gruenehammer-autofreie-innenstadt-ist-wichtiger-erfolg-fuer-die-menschen-in-wien`, Last accessed on 2020-07-09.

[11] Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. Networks, 36(2):69–79.

[12] Babonneau, F., du Merle, O., and Vial, J.-P. (2006). Solving large-scale linear multicommodity flow problems with an active set strategy and proximal-accpm. Operations Research, 54(1):184–197.

[13] Bacci, T., Mattia, S., and Ventura, P. (2020). A branch-and-cut algorithm for the restricted block relocation problem. European Journal of Operational Research.

[14] Baita, F., Pesenti, R., Ukovich, W., and Favaretto, D. (2000). A comparison of different solution approaches to the vehicle scheduling problem in a practical case. Computers & Operations Research, 27:1249–1269.

[15] Baldacci, R., Maniezzo, V., and Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. Operations Research, 52(3):422–439.

[16] Barnhart, C., Hane, C., Johnson, E., and Sigismondi, G. (1994). A column generation and partitioning approach for multi-commodity flow problems. Telecommunication Systems, 3:239–258.

[17] Barnhart, C., Hane, C. A., and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Operations Research, 48(2):318–326.

[18] Barnhart, C., Krishnan, N., and Vance, P. H. (2001). Multicommodity flow problemsmulticommodity flow problems. In Floudas, C. A. and Pardalos, P. M., editors, Encyclopedia of Optimization, pages 1583–1591. Springer US, Boston, MA.

[19] Beermann, M., Jungmeier, G., Wenzel, A., Spitzer, J., Canella, L., Engel, A., Schmuck, M., and Koller, S. (2010). Quo vadis elektroauto? grundlagen einer road map für die einführung von elektro-fahrzeugen in Österreich.

[20] Ben Ticha, H., Absi, N., Feillet, D., and Quilliot, A. (2017). Empirical analysis for the VRPTW with a multigraph representation for the road network. Computers and Operations Research, 88:103–116.

[21] Ben Ticha, H., Absi, N., Feillet, D., and Quilliot, A. (2018). Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows. Computers & Operations Research, 104.

[22] Ben Ticha, H., Absi, N., Feillet, D., Quilliot, A., and van Woensel, T. (2019). A branch-and-price algorithm for the vehicle routing problem with time windows on a road network. Networks.

[23] Bergmann, F. M., Wagner, S. M., and Winkenbach, M. (2020). Integrating first-mile pickup and last-mile delivery on shared vehicle routes for efficient urban e-commerce distribution. Transportation Research Part B: Methodological, 131:26 – 62.

[24] Bertossi, A., Cararesi, P., and Gallo, G. (1987). On some matching problems arising in vehicle scheduling problems. Networks, pages 271–281.

[25] Bérubé, J.-F., Gendreau, M., and Potvin, J.-Y. (2009). An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. European journal of operational research, 194(1):39–50.

[26] Bit-Monnot, A., Artigues, C., Huguet, M.-J., and Killijian, M.-O. (2013). Carpooling: the 2 synchronization points shortest paths problem. In 13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS).

[27] bmvit (2017). `https://www.bmvit.gv.at/verkehr/gesamtverkehr/statistik/downloads/viz_2011_gesamtbericht_270613.pdf`. Last accessed 2017-08-22.

[28] Bodin, L. and Golden, B. (1981). Classification in vehicle routing and scheduling. Networks, 11(2):97–108.

[29] Bodin, L., Golden, B., Assad, A., and Ball, M. (1983). Routing and scheduling of vehicles and crews: The state of the art. Computers & Operations Research, 10(2):63–211. Routing and Scheduling of Vehicles and Crews. The State of the Art.

[30] Boland, N., Charkhgard, H., and Savelsbergh, M. (2015). A criterion space search algorithm for biobjective integer programming: The balanced box method. INFORMS Journal on Computing, 27(4):735–754.

[31] Boyacı, B. and Zografos, K. G. (2019). Investigating the effect of temporal and spatial flexibility on the performance of one-way electric carsharing systems. Transportation Research Part B: Methodological, 129:244 – 272.

[32] Boyacı, B., Zografos, K. G., and Geroliminis, N. (2015). An optimization framework for the development of efficient one-way car-sharing systems. European Journal of Operational Research, 240(3):718 – 733.

[33] Braekers, K., Hartl, R. F., Parragh, S. N., and Tricoire, F. (2016). A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience. European Journal of Operational Research, 248(2):428 – 443.

[34] Brandstätter, G., Gambella, C., Leitner, M., Malaguti, E., Masini, F., Puchinger, J., Ruthmair, M., and Vigo, D. (2016). Overview of Optimization Problems in Electric Car-Sharing System Design and Management. In Dawid, H., Doerner, K. F., Feichtinger, G., Kort, P. M., and Seidl, A., editors, Dynamic Perspectives on Managerial Decision Making, Dynamic Modeling and Econometrics in Economics and Finance, pages 441–471. Springer.

[35] Brandstätter, G., Kahr, M., and Leitner, M. (2017). Determining optimal locations for charging stations of electric car-sharing systems under stochastic demand. Transportation Research Part B: Methodological, 104:17 – 35.

[36] Brandstätter, G., Leitner, M., and Ljubić, I. (2020). Location of charging stations in electric car sharing systems. Transportation Science, 54(5):1153–1438.

[37] Brunsch, T., Cornelissen, K., Manthey, B., and Röglin, H. (2013). Smoothed analysis of the successive shortest path algorithm. In Khanna, S., editor, Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms, pages 1180–1189. SIAM.

[38] Bundesverband CarSharing e.V. (2020). CarSharing in Deutschland. https://www.carsharing.de/sites/default/files/uploads/infografiken_jahresbericht_2020-02.jpg, Last accessed on 2020-05-28.

[39] Bunte, S. and Kliewer, N. (2009). An overview on vehicle scheduling models. Public Transport, 1(4):299–317.

[40] Bünnagel, U., Korte, B., and Vygen, J. (1998). Efficient implementation of the Goldberg–Tarjan minimum-cost flow algorithm. Optimization Methods and Software, 10(2):157–174.

[41] Calvo, R. W., de Luigi, F., Haastrup, P., and Maniezzo, V. (2004). A distributed geographic information system for the daily car pooling problem. Computers & Operations Research, 31(13):2263–2278.

[42] Caramia, M. and Guerriero, F. (2009). A heuristic approach to long-haul freight transportation with multiple objective functions. Omega, 37(3):600–614.

[43] Cattaruzza, D., Absi, N., and Feillet, D. (2016). Vehicle routing problems with multiple trips. 4OR, 14(3):223–259.

[44] Chen, W., Mes, M., Schutten, M., and Quint, J. (2019). A ride-sharing problem with meeting points and return restrictions. Transportation Science, 53(2):401–426.

[45] Ciari, F., Bock, B., and Balmer, M. (2014). Modeling station-based and free-floating carsharing demand: test case study for Berlin. Transportation Research Record, 2416(1):37–47.

[46] CIVITAS (2020). Eccentric. https://civitas.eu/eccentric, Last accessed on 2020-05-27.

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

[48] Dantzig, G. (1963a). Linear programming and extensions. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ.

[49] Dantzig, G. (1963b). Linear programming and extensions. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ.

[50] de Almeida Correia, G. H. and Antunes, A. P. (2012a). Optimization approach to depot location and trip selection in one-way carsharing systems. Transportation Research Part E: Logistics and Transportation Review, 48(1):233 – 247. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

[51] de Almeida Correia, G. H. and Antunes, A. P. (2012b). Optimization approach to depot location and trip selection in one-way carsharing systems. Transportation Research Part E: Logistics and Transportation Review, 48(1):233–247.

[52] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, Parallel Problem Solving from Nature PPSN VI, pages 849–858, Berlin, Heidelberg. Springer Berlin Heidelberg.

[53] Demir, E., Bektaş, T., and Laporte, G. (2014). The bi-objective pollution-routing problem. European Journal of Operational Research, 232(3):464–478.

[54] Desfontaines, L. and Desaulniers, G. (2018). Multiple depot vehicle scheduling with controlled trip shifting. Transportation Research Part B: Methodological, 113:34 – 53.

[55] Doppstadt, C., Koberstein, A., and Vigo, D. (2016). The hybrid electric vehicle – traveling salesman problem. European Journal of Operational Research, 253(3):825 – 842.

[56] d'Orey, P. M. and Ferreira, M. (2014). Can ride-sharing become attractive? A case study of taxi-sharing employing a simulation modelling approach. IET Intelligent Transport Systems, 9(2):210–220.

[57] Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM, 19(2):248–264.

[58] Ehrgott, M. and Gandibleux, X. (2003). Multiobjective Combinatorial Optimization — Theory, Methodology, and Applications, page vol 52. Springer, Boston, MA.

[59] Ehrgott, M., Ljubić, I., and Parragh, S. N. (2017). Feature cluster: Recent advances in exact methods for multi-objective optimisation. European Journal of Operational Research, 260(3):805 – 806.

[60] Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering, 57(4):1472 – 1483.

[61] Enzi, M., Parragh, S. N., and Pisinger, D. (2020a). Modeling and solving a vehicle-sharing problem. arXiv:2003.08207.

[62] Enzi, M., Parragh, S. N., Pisinger, D., and Prandtstetter, M. (2020b). Modeling and solving the multimodal car- and ride-sharing problem. arXiv:2001.05490.

[63] Ervolina, T. R. and McCormick, S. (1993). Two strongly polynomial cut cancelling algorithms for minimum cost network flow. Discrete Applied Mathematics, 46(2):133 – 165.

[64] Eskandarpour, M., Ouelhadj, D., Hatami, S., Juan, A. A., and Khosravi, B. (2019). Enhanced multi-directional local search for the bi-objective heterogeneous vehicle routing problem with multiple driving ranges. European Journal of Operational Research, 277(2):479 – 491.

[65] European Commission (2016). Transport Emissions - A European Strategy for low-emission mobility. https://ec.europa.eu/clima/policies/transport_en, Last accessed on 2018-10-01.

[66] Fahrrad Wien (2020). Radfahren in Zahlen. https://www.fahrradwien.at/radfahren-in-zahlen/radzahlen-2019/, Last accessed on 2020-05-27.

[67] Ferrero, F., Perboli, G., Rosano, M., and Vesco, A. (2018). Car-sharing services: An annotated review. Sustainable Cities and Society, 37:501 – 518.

[68] Ford, L. R. and Fulkerson, D. R. (1962). Flows in Networks. Princeton University Press.

[69] Garaix, T., Artigues, C., Feillet, D., and Josselin, D. (2010). Vehicle routing problems with alternative paths: An application to on-demand transportation. European Journal of Operational Research, 204(1):62 – 75.

[70] Gendreau, M., Ghiani, G., and Guerriero, E. (2015). Time-dependent routing problems: A review. Computers & Operations Research, 64:189 – 197.

[71] Ghannadpour, S. F. and Zarrabi, A. (2019). Multi-objective heterogeneous vehicle routing and scheduling problem with energy minimizing. Swarm and Evolutionary Computation, 44:728 – 747.

[72] Gharari, R., Poursalehi, N., Abbasi, M., and Aghaie, M. (2016). Implementation of strength pareto evolutionary algorithm ii in the multiobjective burnable poison placement optimization of kwu pressurized water reactor. Nuclear Engineering and Technology, 48(5):1126 – 1139.

[73] Goldberg, A. and Tarjan, R. (1989). Finding minimum-cost circulations by canceling negative cycles. Journal of the ACM, 36(4):873–886.

[74] Goldberg, A. V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. Journal of Algorithms, 22(1):1 – 29.

[75] Goldberg, A. V. and Tarjan, R. E. (1990). Finding minimum-cost circulations by successive approximation. Mathematics of Operations Research, 15(3):430–466.

[76] Govindan, K., Jafarian, A., and Nourbakhsh, V. (2019). Designing a sustainable supply chain network integrated with vehicle routing: A comparison of hybrid swarm intelligence metaheuristics. Computers & Operations Research, 110:220 – 235.

[77] Grabenschweiger, J., Tricoire, F., and Doerner, K. F. (2018). Finding the trade-off between emissions and disturbance in an urban context. Flexible Services and Manufacturing Journal, 30(3):554–591.

[78] Groiez, M., Desaulniers, G., Hadjar, A., and Marcotte, O. (2013). Separating valid odd-cycle and odd-set inequalities for the multiple depot vehicle scheduling problem. EURO Journal on Computational Optimization, 1(3):283–312.

[79] Guedes, P. C. and Borenstein, D. (2015). Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. Computers & Industrial Engineering, 90:361 – 370.

[80] Guedes, P. C., Lopes, W. P., Rohde, L. R., and Borenstein, D. (2016). Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem. Optimization Letters, 10(7):1449–1461.

[81] Gunawan, A., Lau, H. C., and Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research, 255(2):315 – 332.

[82] Hadjar, A., Marcotte, O., and Soumis, F. (2006). A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Operations Research, 54(1):130–149.

[83] Hillier, F. S. and Lieberman, G. J. (2001). Introduction to Operations Research. McGraw-Hill, New York, NY, USA, seventh edition.

[84] Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., and Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B: Methodological, 111:395 – 421.

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

[86] Huang, C., Zhang, D., Si, Y.-W., and Leung, S. C. H. (2016). Tabu search for the real-world carpooling problem. Journal of Combinatorial Optimization, 32(2):492–512.

[87] Huang, Y., Zhao, L., van Woensel, T., and Gross, J.-P. (2017). Time-dependent vehicle routing problem with path flexibility. Transportation Research. Part B: Methodological, 95:169–175.

[88] Jorge, D. and Correia, G. H. d. A. (2013). Carsharing systems demand estimation and defined operations: A literature review. European Journal of Transport and Infrastructure Research, 13:201–220.

[89] Jozefowiez, N., Semet, F., and Talbi, E.-G. (2008). Multi-objective vehicle routing problems. European Journal of Operational Research, 189(2):293 – 309.

[90] Karakostas, G. (2008). Faster approximation schemes for fractional multicommodity flow problems. ACM Transactions on Algorithms, 4.

[91] Kek, A. G. H., Cheu, R. L., and Chor, M. L. (2006). Relocation simulation model for multiple-station shared-use vehicle systems. Transportation Research Record, 1986(1):81–88.

[92] Kelly, D., Comm, B. R. P., and O'Neill, G. M. (1991). The minimum cost flow problem and the network simplex solution method.

[93] Klein, M. (1967). A Primal Method for Minimal Cost Flows with Applications to the Assignment and Transportation Problems. Management Science, 14(3):205–220.

[94] Knapen, L., Yasar, A., Cho, S., Keren, D., Dbai, A. A., Bellemans, T., Janssens, D., Wets, G., Schuster, A., Sharfman, I., and Bhaduri, K. (2014). Exploiting graph-theoretic tools for matching in carpooling applications. Journal of Ambient Intelligence and Humanized Computing, 5(3):393–407.

[95] Knopp, S., Biesinger, S., and Prandtstetter, M. (2018). A resource allocation based approach for corporate mobility as a service. Technical report, ARXIV, arXiv:1810.05659.

[96] Kovács, P. (2015). Minimum-cost flow algorithms: an experimental evaluation. Optimization Methods and Software, 30(1):94–127.

[97] Kulkarni, S., Krishnamoorthy, M., Ranade, A., Ernst, A. T., and Patil, R. (2018). A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem. Transportation Research Part B: Methodological, 118:457 – 487.

[98] Laporte, G., Meunier, F., and Wolfler Calvo, R. (2018). Shared mobility systems: an updated survey. Annals of Operations Research, 271(1):105–126.

[99] Li, B., Krushinsky, D., Reijers, H. A., and Woensel, T. V. (2014). The share-a-ride problem: People and parcels sharing taxis. European Journal of Operational Research, 238(1):31 – 40.

[100] Löbel, A. (1996). Solving large-scale real-world minimum-cost flow problems by a network simplex method. Technical Report SC-96-07, ZIB, Takustr. 7, 14195 Berlin.

[101] Lübbecke, M. E. (2011). Column Generation. American Cancer Society.

[102] MA 18 - Stadtentwicklung und Stadtplanung Wien (2015). Carsharing Wien Evaluierung. `https://www.wien.gv.at/stadtentwicklung/studien/pdf/b008470.pdf`, Last accessed on 2020-05-28.

[103] MA 46 (2020). Entwicklung des Radverkehrsnetzes in Wien (2000-2019). `https://www.wien.gv.at/verkehr/radfahren/pdf/fakten-1.pdf`, Last accessed on 2020-05-27.

[104] Maciejewski, M., Salanova, J. M., Bischoff, J., and Estrada, M. (2016). Large-scale microscopic simulation of taxi services. berlin and barcelona case studies. Journal of Ambient Intelligence and Humanized Computing, 7(3):385–393.

[105] Martin Kords (2020). Pkw-Bestand in Wien bis 2019. `https://de.statista.com/statistik/daten/studie/683923/umfrage/pkw-bestand-in-wien/`, Last accessed on 2020-05-27.

[106] Martínez-Salazar, I., Ángel Bello, F., and Alvarez, A. (2015). A customer-centric routing problem with multiple trips of a single vehicle. Journal of the Operational Research Society, 66.

[107] Masoud, N. and Jayakrishnan, R. (2017). A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem. Transportation Research Part B: Methodological, 99:1–29.

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

[109] Matl, P., Hartl, R. F., and Vidal, T. (2019). Leveraging single-objective heuristics to solve bi-objective problems: Heuristic box splitting and its application to vehicle routing. Networks, 73(4):382–400.

[110] Mitchell, J. E. (2002). Branch-and-cut algorithms for combinatorial optimization problems. Handbook of applied optimization, 1:65–77.

[111] Mourad, A., Puchinger, J., and Chu, C. (2019a). A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological, 123:323 – 346.

[112] Mourad, A., Puchinger, J., and Chu, C. (2019b). A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological, 123:323 – 346.

[113] Mulley, C., Nelson, J. D., and Wright, S. (2018). Community transport meets mobility as a service: On the road to a new a flexible future. Research in Transportation Economics.

[114] Nair, R. and Miller-Hooks, E. (2014). Equilibrium network design of shared-vehicle systems. European Journal of Operational Research, 235(1):47–61.

[115] Ngueveu, S. U., Prins, C., and Calvo, R. W. (2010). A Hybrid Tabu Search for the m-Peripatetic Vehicle Routing Problem, pages 253–266. Springer US, Boston, MA.

[116] Nolz, P. C., Absi, N., and Feillet, D. (2014). A bi-objective inventory routing problem for sustainable waste management under uncertainty. Journal of Multi-Criteria Decision Analysis, 21(5-6):299–314.

[117] Oukil, A., Amor, H. B., Desrosiers, J., and Gueddari, H. E. (2007). Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. Computers & Operations Research, 34(3):817 – 834. Logistics of Health Care Management.

[118] Paris en Selle (2020). Plan vélo. `https://planvelo.paris/`, Last accessed on 2020-05-27.

[119] Parragh, S. N. and Tricoire, F. (2019). Branch-and-bound for bi-objective integer programming. INFORMS Journal on Computing, 31(4):805–822.

[120] Pepin, A.-S., Desaulniers, G., Hertz, A., and Huisman, D. (2008). A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling, 12(1):17.

[121] Qu, Y. and Bard, J. F. (2012). A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. Computers & Operations Research, 39(10):2439 – 2456.

[122] Regue, R., Masoud, N., and Recker, W. (2016). Car2work. Transportation Research Record: Journal of the Transportation Research Board, 2542:102–110.

[123] Repoux, M., Kaspi, M., Boyacı, B., and Geroliminis, N. (2019). Dynamic prediction-based relocation policies in one-way station-based carsharing systems with complete journey reservations. Transportation Research Part B: Methodological, 130:82 – 104.

[124] Rétvári, G., Bíró, J., and Cinkler, T. (2004). A novel lagrangian-relaxation to the minimum cost multicommodity flow problem and its application to ospf traffic engineering. Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No.04TH8769), 2:957–962 Vol.2.

[125] Ribeiro, C. C. and Soumis, F. (1994). A column generation approach to the multiple-depot vehicle scheduling problem. Operations Research, 42(1):41–52.

[126] Riera-Ledesma, J. and Salazar-González, J. J. (2005). The biobjective travelling purchaser problem. European Journal of Operational Research, 160(3):599 – 613. Decision Analysis and Artificial Intelligence.

[127] SEAMLESS (2020). SEAMLESS - Sustainable, Efficient Austrian Mobility with Low-Emission Shared Systems. http://www.seamless-project.at/projekt/, Last accessed on 2020-06-02.

[128] Silva, M. M., Subramanian, A., Vidal, T., and Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. European Journal of Operational Research, 221(3):513 – 520.

[129] Sioui, L., Morency, C., and Trépanier, M. (2013). How carsharing affects the travel behavior of households: a case study of montréal, canada. International journal of sustainable transportation, 7(1):52–69.

[130] Speranza, M., Archetti, C., and Vigo, D. (2014). Chapter 10: Vehicle Routing Problems with Profits.

[131] Srinivasan, V. and Thompson, G. L. (1976). Algorithms for minimizing total cost, bottleneck time and bottleneck shipment in transportation problems. Naval Research Logistics Quarterly, 23(4):567–595.

[132] Stadt Wien (2017). https://www.wien.gv.at/stadtentwicklung/studien/pdf/b008404.pdf, Last accessed on 2017-08-22.

[133] Stadtentwicklung Wien (2020). Step2025 stadtentwicklungsplan wien. https://www.wien.gv.at/stadtentwicklung/studien/pdf/b008379a.pdf, Last accessed on 2020-05-27.

[134] Statistik Austria (2017). http://www.statistik.at/web_de/statistiken/menschen_und_gesellschaft/arbeitsmarkt/erwerbstaetige/062875.html, Last accessed on 2017-08-22.

[135] Statistik Wien (2020). Wiener Bevölkerungsstand. https://www.wien.gv.at/statistik/bevoelkerung/bevoelkerungsstand/index.html, Last accessed on 2020-05-27.

[136] Stidsen, T., Andersen, K. A., and Dammann, B. (2014). A branch and bound algorithm for a class of biobjective mixed integer programs. Management Science, 60(4):1009–1032.

[137] Tachet, R., Sagarra, O., Santi, P., Resta, G., Szell, M., Strogatz, S., and Ratti, C. (2017). Scaling law of urban ride sharing. Scientific reports, 7:42868.

[138] Tardos, E. (1985). A strongly polynomial minimum cost circulation algorithm. Combinatorica, 5:247–256.

[139] Tomlin, J. A. (1966). Minimum-cost multicommodity network flows. Operations Research, 14(1):45–51.

[140] Toro, E. M., Franco, J. F., Echeverri, M. G., and Guimarães, F. G. (2017). A multi-objective model for the green capacitated location-routing problem considering environmental impact. Computers & Industrial Engineering, 110:114 – 125.

[141] Toth, P. and Vigo, D. (2002). The Vehicle Routing Problem. Society for Industrial and Applied Mathematics.

[142] Tricoire, F., Graf, A., and Gutjahr, W. J. (2012). The bi-objective stochastic covering tour problem. Computers & Operations Research, 39(7):1582 – 1592.

[143] Tricoire, F. and Parragh, S. N. (2017). Investing in logistics facilities today to reduce routing emissions tomorrow. Transportation Research Part B: Methodological, 103:56 – 67. Green Urban Transportation.

[144] United Nations - Department of Economic and Social Affairs (2018). 68% of the world population projected to live in urban areas by 2050, says un. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html, Last accessed on 2020-05-28.

[145] VCÖ - Mobilität der Zukunft (2020). VCÖ-Factsheet 2018-10 - Großes Potenzial für Sharing und neue Mobilitätsservices. https://www.vcoe.at/files/vcoe/uploads/News/VCOe-Factsheets/2018/2018-10%20Sharing%20und%20neue%20Mobilitaetsloesungen/VCO%CC%88-Factsheet%20Sharing%20und%20neue%20Mobilita%CC%88tslo%CC%88sungen.pdf, Last accessed on 2020-05-27.

[146] Vidal, T., Laporte, G., and Matl, P. (2019). A concise guide to existing and emerging vehicle routing problem variants. CoRR, abs/1906.06750.

[147] Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., and Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. Computers & Operations Research, 40(1):498 – 509.

[148] Visée, M., Teghem, J., Pirlot, M., and Ulungu, E. (1998). Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem. Journal of Global Optimization, 12:139 – 155.

[149] Weikl, S. and Bogenberger, K. (2013). Relocation strategies and algorithms for free-floating car sharing systems. IEEE Intelligent Transportation Systems Magazine, 5(4):100–111.

[150] Wiener Linien (2010). Wahl der Verkehrsmittel und Anteil des öffentlichen Verkehrs. https://www.wien.gv.at/presse/bilder/2011/02/15/839-mio-fahrgaeste-fahrgastrekord-2010-fuer-die-wiener-linien, Last accessed on 2020-05-27.

[151] Wiener Linien (2019). Modal Split 2019. https://www.wien.gv.at/presse/bilder/2020/02/12/modal-split-2019_wiener-linien-png, Last accessed on 2020-05-27.

[152] Wiener Linien (2020). Die wiener "Öffis" in zahlen. https://www.wienerlinien.at/eportal3/ep/contentView.do/pageTypeId/66528/programId/67199/contentTypeId/1001/channelId/-47395/contentId/68061, Last accessed on 2020-07-09.

[153] YV, Y. H., Lasdon, L. S., and Da Wismer, D. (1971). On a bicriterion formation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man and Cybernetics, (3):296–297.

[154] Zhang, D., Liu, Y., and He, S. (2019). Vehicle assignment and relays for one-way electric car-sharing systems. Transportation Research Part B: Methodological, 120:125 – 146.

[155] Zhou, B. and Kockelman, K. M. (2011). Opportunities for and impacts of car-sharing: A survey of the austin, texas market. International Journal of Sustainable Transportation, 5(3):135–152.

[156] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation, 3(4):257–271.

[157] Österreich, B. (2020). Aus Verantwortung für Österreich. Regierungsprogramm 2020–2024.

Abstract

The way we use and see mobility is changing. Especially in urban areas, the people tend to seize the multimodal character of a city. Some always take a bike, others prefer public transportation, others prefer taxis or personal cars. Nevertheless, the latter one is decreasing in its importance and as the prevailing mode of transport. Many users switch to more environmentally friendly modes of transport and/or use sharing systems. We can see that cities prominently state their mobility plans and concepts. This trend is not only visible in our private lives but increasingly important in a corporate context. Thus, companies aim to have an overall green and sustainable profile. Instead of supporting further developments in corporate mobility privileging a few selected users, we aim at providing sustainable corporate mobility concepts, inspired by the changes also seen in our private environment.

In this thesis we model shared mobility systems for a closed group of users, and propose efficient solution techniques to solve complex problems. We first introduce the vehicle-sharing problem where one or multiple types of shared vehicles may be shared. We model it as a minimum-cost flow problem and a multi-commodity flow problem. A thorough discussion gives insights into managerial implications regarding the employment of sharing systems in a company. Then, secondly, we extend the idea of vehicle-sharing by allowing users to co-ride with each other and introduce the multimodal car- and ride-sharing problem. We employ a two-layer decomposition approach, where the first layer is solved by complete enumeration and the second layer by a column generation approach to find car routes through a day. We solve realistic instances in reasonable time, making a daily planning of such car- and ride-sharing system possible. Thirdly, we introduce the bi-objective multimodal car-sharing problem where we include the user preferences in a second objective. We study four variants of the problem and solve the most challenging one with a branch-and-cut approach.

All models and algorithms are tested using generated data based on gathered statistics considering Viennese work patterns. Throughout the thesis we provide holistic algorithmic tests but also socio-economic insights and managerial implications.

Zusammenfassung

Das Mobilitätsverhalten - insbesondere in Städten - verändert sich. Während vor wenigen Jahren das Privatauto das vorherrschende Verkehrsmittel war, wird heutzutage die vielfältige Infrastruktur in Städten ausgiebig genutzt. Die Anbindung der öffentlichen Verkehrsmittel wird zunehmend gerne genutzt, das Fahrrad wird dem Auto immer öfter vorgezogen, oder es wird auf Sharing-Systeme in Kombination mit anderen Verkehrsmitteln zurückgegriffen. Der Wandel wird durch Großprojekte in diversen europäischen Städten unterstützt. Dabei wird weitgehend auf "grüne" Verkehrsmittel gesetzt, Parkplätze werden zu Nutzflächen für Stadtbewohnerinnen und Stadtbewohner. Die Entwicklungen wirken sich sowohl auf das private, wie auch auf das unternehmerische Mobilitätsverhalten und die angebotenen Möglichkeiten im jeweiligen Bereich aus.

Die vorliegende Dissertation beschäftigt sich mit der Modellierung von Shared-Mobilitätssystemen in Firmen und führt effiziente Lösungsverfahren solch komplexer Probleme ein. Es werden drei Modelle diskutiert: (1) das Vehicle-Sharing Problem, (2) das Multimodal Car- and Ride-Sharing Problem, sowie (3) das Bi-Objective Multimodal Car-Sharing Problem. Wir beginnen mit einem System, in dem Fahrzeuge gemeinschaftlich genutzt werden und zielen dabei auf optimale Ressourcennutzung ab. Das Vehicle-Sharing Problem wird mittels bewährter und effizienter Formulierungen modelliert. Das Konzept des reinen Car-Sharings wird mit der Idee einer Fahrgemeinschaft innerhalb eines Unternehmens im Multimodal Car- and Ride-Sharing Problem erweitert und mittels Column Generation gelöst. Des Weiteren werden Präferenzen der User in einer zusätzliche Zielfunktion berücksichtig und somit ein Zweiziel-Problem aufgebaut. Wir diskutieren unterschiedliche Varianten des Bi-Objective Multimodal Car-Sharing Problems und nehmen, zur Lösung der herausforderndsten Variante, einen Branch-And-Cut Algorithmus zur Hand.

Die Modelle und Algorithmen werden mit realistischen Instanzen getestet. Die Daten basieren auf Statistiken zum Arbeits- und Mobilitätsverhalten am Beispiel der Stadt Wien. Die einzelnen Kapitel beinhalten nicht nur ausführliche algorithmische Tests, sondern auch umfassende Diskussionen bezüglich unternehmerischer Auswirkungen und gibt darüber hinaus sozio-ökonomische Einblicke.