The bi-objective multimodal car-sharing problem

Miriam Enzi, Sophie N. Parragh, Jakob Puchinger

OR Spectrum (2022) 44:307–348
https://doi.org/10.1007/s00291-021-00631-2

ORIGINAL ARTICLE

The bi-objective multimodal car-sharing problem

Miriam Enzi[1,2] · **Sophie N. Parragh**[1] · **Jakob Puchinger**[3,4]

Received: 23 June 2020 / Accepted: 11 April 2021 / Published online: 14 September 2021

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 fulfil 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

✉ Miriam Enzi
miriam.enzi@gmx.net

Sophie N. Parragh
sophie.parragh@jku.at

Jakob Puchinger
jakob.puchinger@irt-systemx.fr

[1] Institute of Production and Logistics Management, Johannes Kepler University Linz, Linz, Austria

[2] Center for Energy, AIT Austrian Institute of Technology, Vienna, Austria

[3] Laboratoire Génie Industriel, CentraleSupélec, Université Paris-Saclay , Gif-sur-Yvette, France

[4] Institut de Recherche Technologique SystemX, Palaiseau, France

1 Introduction

Today, most of the world's population lives in urban environments and cities continue to grow (United Nations-Department of Economic and Social Affairs 2018). 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, etc.). 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. (2019) 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 (VCÖ - Mobilität der Zukunft 2020). In Vienna (Austria), the number of cars per capita is constantly decreasing (Martin 2020; Statistik Wien 2020). 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 (Wiener 2010, 2019). Thus, people move to alternative, more environmentally friendly MOTs.

Additionally, citizens increasingly use sharing systems (VCÖ - Mobilität der Zukunft 2020). In Germany, the number of shared cars has increased fivefold within ten years; there are almost twelve times more users than a decade ago (Bundesverband CarSharing eV 2020). In Vienna (Austria), one shared car eliminates the need of five privately owned ones (MA 18 - Stadtentwicklung und Stadtplanung Wien 2015). 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 (VCÖ - Mobilität der Zukunft 2020). Moreover, car-sharing saves up to 44 million car-kilometres in Vienna annually. This equals to approximately 7000 tons of CO_2 (MA 18 - Stadtentwicklung und Stadtplanung Wien 2015). 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 (VCÖ - Mobilität der Zukunft 2020).

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' (Stadtentwicklung Wien 2020). Madrid is aiming to establish a holistic 'Mobility as a Service' concept offering real-time information and including over 30 shared mobility options (CIVITAS 202 2020). 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 kilometres driven on specific legs (Fahrrad 2020; MA 46 2020). 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 (Paris en Selle 2020).

Novel mobility concepts and reconsidering mobility play 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 (SEAMLESS 2020).

Traveller experience needs to be taken into account in the design of novel mobility systems and is key to its success with users (Al Maghraoui et al. 2019). 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 (Ferrero et al. 2018), 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 in 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 Sect. 2, we review related work. Section 3 introduces the BiO-MMCP where Sect. 3.1 gives a problem formulation, followed by the formal description in Sect. 3.2 for all four variants. In Sect. 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 Sect. 4.2. Moreover, we introduce a set of valid inequalities in Sect. 4.1 and describe the bi-objective frameworks used in this paper in Sect. 4.3. Section 5 summarizes our computational study. Finally, we draw conclusions and we give an outlook to future work in Sect. 6.

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. (2018).

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 (Zhou and Kockelman 2011; Sioui et al. 2013). The potential and effects of such systems are then often determined through simulation-based approaches (Ciari et al. 2014). From an operational perspective, problems considered in the car-sharing-related literature are mainly concerned with relocating, recharging and servicing vehicles (de Almeida Correia and Antunes 2012; Nair and Miller-Hooks 2014; Weikl and Bogenberger 2013). 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 analysing 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. (2019). 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 (Calvo et al. 2004; d'Orey and Ferreira 2014; Maciejewski et al. 2016; Tachet et al. 2017).

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. (2010) 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. (2017) 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 (Toth and Vigo 2002; Eksioglu et al. 2009) in our work. Our paper introduces a kind of multi-trip VRP (Cattaruzza et al. 2016) with heterogeneous vehicles and multiple depots on a multi graph. Garaix et al. (2010) were among the first who studied VRPs with alternative arcs between each pair of nodes. VRPs with multiple attributes (Garaix et al. 2010) or multi-graphs in the VRP stream have gained increasing attention in the past years (Doppstadt et al. 2016; Ben Ticha et al. 2017, 2018, 2019; Huang et al. 2017), whereas, of course, mutlimodality significantly enlarges the set of possible solutions (Caramia and Guerriero 2009). 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. (2015), 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. (2019), a detailed analysis through VRP variants also tackling customer-centred objectives is provided. As an example, the cumulative VRP (Ngueveu et al. 2010; Silva et al. 2012) replaces the classical minimum cost objective function with the minimization of individual customer arrival times. Martínez-Salazar et al. (2015) 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. (2016) 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. (2008) 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 (Abad et al. 2018; Alexiou and Katsavounis 2015; Androutsopoulos and Zografos 2017; Demir et al. 2014; Eskandarpour et al. 2019; Ghannadpour and Zarrabi 2019; Toro et al. 2017; Tricoire and Parragh 2017; Govindan et al. 2019; Anderluh

et al. 2019; Grabenschweiger et al. 2018) or external social criteria (Ghannadpour and Zarrabi 2019; Govindan et al. 2019; Nolz et al. 2014; Anderluh et al. 2019; Grabenschweiger et al. 2018) 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 (Deb et al. 2000) or the SPEA-II (Gharari et al. 2016). However, only recently Matl et al. (2019) have shown that single-objective VRP heuristics can be efficiently used in an ϵ-constrained-based method. The ϵ-constraint method (Yh et al. 1971; Srinivasan and Thompson 1976) 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 (Visée et al. 1998), the weighted sum approach (Aneja and Nair 1979) or, more recently, the balanced box method (Boland et al. 2015) and the weighting binary search method (Riera-Ledesma and Salazar-González 2005). 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 (Stidsen et al. 2014; Vincent et al. 2013; Parragh and Tricoire 2019; Adelgren and Gupte 2017). A recent overview of exact methods for multi-objective optimization is provided in Ehrgott et al. (2017). A detailed overview of general multi-objective combinatorial optimization is provided by Ehrgott and Gandibleux (2003). 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 (2005), who developed a weighting method and conduct a binary search in the objective space. Moreover, similar to Bérubé et al. (2009), 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 (2005), cuts from prior iterations are added to the cut pool for further iterations. Contrary to Riera-Ledesma and Salazar-González (2005) and Bérubé et al. (2009), we add detected cuts as hard constraints, showing better results for our problem setting.

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.

3.1 Problem description

The BiO-MMCP aims to assign modes of transport to user trips and determining car routes during a day whilst 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.

3.2 Formal description

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

Table 1 Mathematical notation used in the formal description of the BiO-MMCP

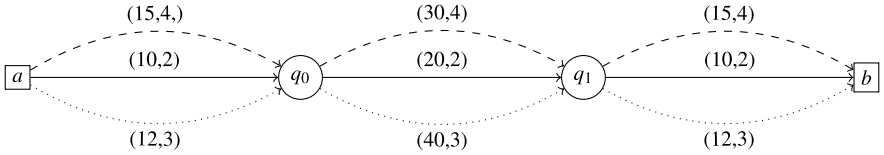
Sets and nodes	
P	Set of users
R	Set of trips
$Q_r \subseteq Q$	Set of tasks of trip r as a subset of the set of tasks
a_r	Start node of a trip r
b_r	End node of a trip r
G_r	Set of nodes on a trip r
D	Set of depots
K	Set of modes of transport
L	Set of legs
L^-, L^+	Set of ingoing/outgoing legs
L_v^-, 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

Table 1 (continued)

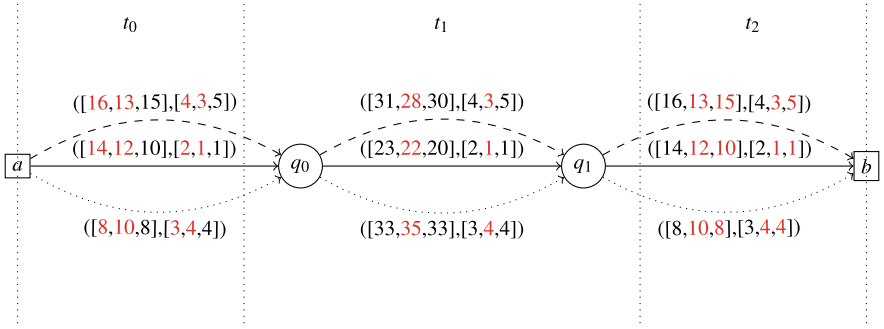
s_v	Service time at node v
o_l	Interval start of leg l
e_l	Interval end of leg l
$\mathcal{W}$	Accumulated waiting time
Δ	Value stating how much a route can be moved forward
F	Forward slack, $F = \mathcal{W} + \Delta$
Decision variables	
x_l	1 if leg l is chosen, 0 otherwise
τ_l	Departure time of leg l

leg l completely lies within a period t the preference value of the leg θ_l equals σ^t_{pk}. 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 1a 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



(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

Fig. 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

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 1b shows the same trip as Fig. 1a, 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.

3.2.1 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.

$$\min \quad \sum_{l \in L} c_l x_l \tag{1}$$

$$\min \quad \sum_{l \in L} \theta_l x_l \tag{2}$$

$$\text{s.t} \sum_{k \in K} \sum_{l \in L^+_{vk}} x_l = 1 \quad \forall v \in V' \tag{3}$$

$$\sum_{l \in L^-_{vk}} x_l = \sum_{l \in L^+_{vk}} x_l \quad \forall v \in V', k \in K \tag{4}$$

$$\sum_{l \in L^+_{dk}} x_l \leq W_{dk} \quad \forall d \in D, k \in K \tag{5}$$

$$\sum_{l \in L^-_{d'k}} x_l \leq \overline{W}_{d'k} \quad \forall d' \in D, k \in K \tag{6}$$

$$x_l \in \{0,1\} \quad \forall l \in L \tag{7}$$

The objective (1) minimizes total cost and objective (2) minimizes user-centred MOT preferences. Constraints (3) make sure that each node v is covered by exactly one leg l. Constraints (4) ensure flow conservation at nodes $v \in V'$ for every MOT k. Constraints (5) and (6) restrict the number of available MOTs W_{dk}, $\overline{W}_{d'k}$ at the start and end of the time horizon. Constraints (7) define the domains of the decision variables.

3.2.2 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.

3.2.3 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 modelled 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 min, 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.

$$\begin{aligned} &\min && (1)\\ &\min && (2)\\ &\text{s.t} && (3)-(7)\\ &&& \tau_l + t_l + s_v - \tau_n \leq M(2 - x_l - x_n) \quad \forall l \in L^-_{vk}, n \in L^+_{vk}, v \in V', k \in K \end{aligned} \tag{8}$$

$$\sum_{l \in L^-_v} (\tau_l + t_l x_l) + s_v \geq \sum_{l \in L^+_v} \tau_l - h \quad \forall v \in V' \tag{9}$$

$$\tau_l \leq H x_l \quad \forall l \in L \tag{10}$$

$$\sum_{l \in L^+_{\gamma_p}} x_l = 1 \quad \forall p \in P \tag{11}$$

$$\sum_{l \in L^-_{\phi_p}} x_l = 1 \quad \forall p \in P \tag{12}$$

$$\sum_{l \in L^-_{vp}} x_l = \sum_{l \in L^+_v} x_l \quad \forall v \in A_p, p \in P \tag{13}$$

$$\sum_{l \in L^-_v} x_l = \sum_{l \in L^+_{vp}} x_l \quad \forall v \in B_p, p \in P \tag{14}$$

$$\tau_l + t_l + s_v - \tau_n \leq M(2 - x_l - x_n) \quad \forall l \in L^-_{vp}, n \in \quad L^+_{vp}, v \in V' \cup \{\gamma_p, \phi_p\}, p \in P \tag{15}$$

$$\tau_l \geq 0 \quad \forall l \in L \tag{16}$$

Constraints (8) set the time variables and take care of subtour elimination within trips. Constraints (9) ensure that a user is leaving at the latest h minutes after the end of the task. Constraints (10) restrict the latest departure time at any task to be at the end of the time horizon. Constraints (11) and (12) make sure that each user is starting her route in node γ_p and ending in node ϕ_p. Constraints (13) and (14) balance the flows of start and end nodes of user p. Constraints (15) eliminate parallel trips. Finally, constraints (16) make sure that decision variables τ are non-negative.

3.2.4 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^t_{pk}.

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:

$$o_l x_l \leq \tau_l \leq e_l \quad \forall l \in L \tag{17}$$

Constraints (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 (8), (9), (10), (15), (16), and (17) by infeasible path constraints (Ascheuer et al. 2000) (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:

$$\begin{aligned} &\min \quad (1) \\ &\min \quad (2) \\ &\text{s.t } (3)-(7), (11)-(14) \\ &\quad \sum_{l \in L(\rho)} x_l \leq |L(\rho)| - 1 \quad \forall \rho \in \mathcal{R}_{car} \end{aligned} \tag{18}$$

$$\sum_{l \in L(\rho)} x_l \leq |L(\rho)| - 1 \quad \forall \rho \in \mathcal{R}_p \tag{19}$$

$$\sum_{l \in L(S)} x_l \leq |S| - 1 \quad \forall S \subseteq G_r, r \in R, S \neq \emptyset \tag{20}$$

Constraints (18)–(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 (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 Solution approach

In the following, we first introduce valid inequalities in Sect. 4.1. By embedding the models into bi-objective optimization frameworks, described in Sect. 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 Sect. 4.2 for model `m4b`.

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:

$$\sum_{l \in L^+_{vk}} x_l + \sum_{g \in K : g \neq k} \sum_{l \in L^-_{vg}} x_l = 1 \quad \forall v \in V', k \in K \tag{21}$$

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:

$$\sum_{k \in K} \sum_{l \in L^-_{vk}} x_l = 1 \quad \forall v \in V' \tag{22}$$

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:

$$\sum_{a \in A} \sum_{l \in L^-_{ak}} x_l \geq \sum_{d \in D} \sum_{g \in L^+_{dk}} x_g \quad \text{with } k = \text{car} \tag{23}$$

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:

$$\sum_{l \in L_p} x_l \geq 4 \quad \forall p \in P \tag{24}$$

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. :

$$\sum_{l \in L_v^- : v \in G_r} x_l \geq |G_r| - 1 \quad \forall r \in R \tag{25}$$

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:

$$\sum_{l \in L_p} x_l = |V_p| - 1 \quad \forall p \in P \tag{26}$$

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:

$$\sum_{l \in L^{(v,v')}} x_l + \sum_{l \in L^{(v',v)}} x_l \leq 1 \quad \forall (v, v') \in L \tag{27}$$

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 have 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:

$$o_l < o_g \wedge e_l < e_g \iff x_l + x_g \leq 1 \quad \forall l \in L_v^+, g \in L_v^-, v \in V' \tag{28}$$

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 (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 legs ($o_g > o_l$,$e_g > e_l$), we can again say that at most one leg can be chosen. Both valid inequalities can be formulated as follows:

$$x_g + \sum_{l \in L_v^+ : o_l < o_g \wedge e_l < e_g} x_l \leq 1 \quad \forall g \in L_v^-, v \in V' \tag{29}$$

$$x_l + \sum_{g \in L_v^- : o_g > o_l \wedge e_g > e_l} x_g \leq 1 \quad \forall l \in L_v^+, v \in V' \tag{30}$$

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:

$$x_g + \sum_{l \in L_v^+ : o_l > e_g + t_g + s_v + h} x_l \leq 1 \quad \forall g \in L_v^-, v \in V' \tag{31}$$

$$x_l + \sum_{g \in L_v^- : o_l > e_g + t_g + s_v + h} x_g \leq 1 \quad \forall l \in L_v^+, v \in V' \tag{32}$$

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 inequalities as follows:

$$x_g + \sum_{l \in L_v^+ : o_g + t_g + s_v > e_l} x_l \leq 1 \quad \forall g \in L_v^-, v \in V' \tag{33}$$

$$x_l + \sum_{g \in L_v^- : o_g + t_g + s_v > e_l} x_g \leq 1 \quad \forall l \in L_v^+, v \in V' \tag{34}$$

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 (18)–(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.

4.2.1 Separation of infeasible user routes

We separate infeasible user routes for each user $p \in P$. Let $\bar{x}$ 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 $\bar{x}_l = 1$ 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 $F, \mathcal{W}, \tau$ 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 Appendix in Algorithm 1.

4.2.2 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 $\bar{x}_l = 1$ and the MOT $k =$ car in the route ρ. Whilst 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 Appendix.

4.2.3 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 $L'^{-}_{vk}, L'^{+}_{vk}, L'^{-}_{vp}, L'^{+}_{vp}$ ($\bar{x}_l = 1$ in the current solution), and we solve the following small LP derived from constraints (8), (9) and (15):

$$\tau_l + t_l + s_v \leq \tau_n \quad \forall l \in L'^{-}_{vk}, n \in L'^{+}_{vk}, v \in V', k \in K \tag{35}$$

$$\tau_l + t_l + s_v \leq \tau_n \quad \forall l \in L'^{-}_{vp}, n \in L'^{+}_{vp}, v \in V' \cup U, p \in P \tag{36}$$

$$\tau_l + t_l + s_v \geq \tau_n - h \quad \forall l \in L'^{-}_{vk}, n \in L'^{+}_{vk}, v \in V', k \in K \tag{37}$$

Constraints (35) and (36) synchronize the car and user route with the decision variable τ. Furthermore, constraints (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:

$$\sum_{l \in L''} x_l \leq |L''| - 1 \tag{38}$$

4.2.4 Strengthened infeasible path constraints

The infeasible paths introduced before in the form of constraint (18) and constraint (19) are very weak. We strengthen them as follows:

$$\sum_{l \in \rho} x_l + \sum_{l \in \mathcal{L}'} x_l + \sum_{l \in \mathcal{L}''} x_l \leq |\rho| - 1 \tag{39}$$

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:

$$\sum_{l \in \rho_{\text{short}}} x_l + \sum_{l \in \mathcal{L}'} x_l + \sum_{l \in \mathcal{L}''} x_l \leq |\rho_{\text{short}}| - 1 \tag{40}$$

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.

4.3.1 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 (1) as the main objective, and the preferences objective (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.

4.3.2 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 (2005). 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 $\left(f_1^{(1)}, f_2^{(1)}\right)$ and $\left(f_1^{(2)}, f_2^{(2)}\right)$. $f_1^{(1)}$ and $f_1^{(2)}$ give the first, e.g. cost, solutions, of the respec-

tive 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 $\left(f_1^{(1)}, f_2^{(1)}\right)$ and $\left(f_1^*, f_2^*\right)$, as well as with $\left(f_1^*, f_2^*\right)$ and $\left(f_1^{(2)}, f_2^{(2)}\right)$. 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.

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 is used.

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. (2020a) and Enzi et al. (2020b). Note that the instances represent a company within a city; thus, the data do 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 are included in Table 10 in Appendix. 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 11 in Appendix). Furthermore, we implement an increase/decrease in cost and time for the respective time periods (see Table 11 in Appendix). 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.

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 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 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.

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

5.3.1 Comparison of models m1, m2, m3, and m4

In a first step, we compare the four models regarding their run times. Table 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

Table 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 $|P| = 20, 50, 100, 150, 200, 250, 300$

	$\|P\|$ =	20	50	100	150	200	250	300
$\|\overline{V'}\|$		95	242	476	714	951	1179	1422
$\|\overline{R}\|$		31	76	147	218	287	358	427
$\|\overline{L}\|$	m1, m2	416	1188	2612	4340	6315	8734	11,038
	m3	947	4190	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	3984	13,995	34,780	61,310	92,790	127,155	167,645

Row 'm4,m4b+GR' gives the average number of legs after the graph reduction

Table 3 Average computational times in seconds for models m1, m2, m3, m3-VI, m4, m4bVIBnCBiO and an increasing number of users $|P| = 20, 50, 100, 150, 200, 250, 300$ for both directions (cost, pref) in the ϵ-constraint method

	$\|P\| =$	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	1037.3	–	–	–	–	–
m3-VI	cost	8.0	480.4	5743.2	–	–	–	–
	pref	8.5	874.5	5577.5	–	–	–	–
m4	cost	–	–	–	–	–	–	–
	pref	–	–	–	–	–	–	–
m4bVIBnCBiO	cost	3511.9	–	–	–	–	–	–
	pref	2730.7	–	–	–	–	–	–

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

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 summarizes the average number of Pareto optimal solutions per instance set. The number of solutions is moderately increasing with number of users $|P|$. Comparing m1 and m3, we see almost the same number of Pareto optimal solutions

Table 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	1073
m3	18	72	158	–	–	–	–
m4	141	–	–	–	–	–	–

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 understanding of the resulting solutions. Comparing m1 with m2, we can see a distinct increase in 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 $|P| = 20$.

5.3.2 Introducing valid inequalities for model m3

We now analyse the impact of the proposed valid inequalities (VI) in more detail. Table 5 presents the computational times in seconds solving m3 without additional information (m3) and by adding valid inequalities (21)–(27) as well as subtour elimination constraints (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,

Table 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

	cost		pref			cost		pref	
	m3	m3-VI	m3	m3-VI		m3	m3-VI	m3	m3-VI
E_100_0	**2419**	2852	3212	2628	E_150_0	TO	TO	TO	TO
E_100_1	3329	3104	**2513**	2777	E_150_1	TO	TO	TO	**39,507**
E_100_2	35,758	**7275**	TO	8504	E_150_2	TO	TO	TO	**41,675**
E_100_3	3382	3544	**3281**	3892	E_150_3	TO	37,060	TO	**20,507**
E_100_4	24,162	**4177**	25,159	4553	E_150_4	TO	TO	TO	TO
E_100_5	27,186	6655	25,587	**6062**	E_150_5	TO	19,766	TO	**14,098**
E_100_6	11,152	9814	11,403	**9092**	E_150_6	TO	18,352	TO	**13,640**
E_100_7	24,112	9245	22,085	**6880**	E_150_7	TO	38,193	TO	**21,112**
E_100_8	3398	5204	**2685**	3678	E_150_8	TO	30,083	TO	**23,643**
E_100_9	22,771	**5561**	28,090	7708	E_150_9	TO	TO	TO	**35,617**
# solved	10	10	9	10		0	5	0	8

Bold values represent the best run time for the respective instance

'# solved' shows the number of instances solved. TO = time out

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.

5.3.3 Solving model m4

We now compare different approaches for solving m4. Table 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 (39)-(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 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 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 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

Table 6 Average computational times in seconds for each instance of $|P| = 20$ solving m4 using different approaches

	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	2957	2099	455	**407**	1823	621
E_20_2	TO	20,437	13,504	9339	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	2447	1798	473	**420**	1681	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	1982	1557	399	**369**	1244	590
E_20_7	TO	TO	TO	TO	3383	2436	628	**496**	2555	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

Bold values represent the best run time for the respective instance

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

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 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 Fig. 2, we show the number of cuts added at each iteration for one chosen instance, namely E_20_0. Figure 2a and b shows 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. Figure 2c and d gives the number of cuts added 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 Fig. 2b 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 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))

	`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	5876	13,954
E_50_5	41,764	TO
E_50_6	TO	TO
E_50_7	TO	TO
E_50_8	4746	7907
E_50_9	TO	TO

Both approaches add prior detected infeasible paths as constraints to model

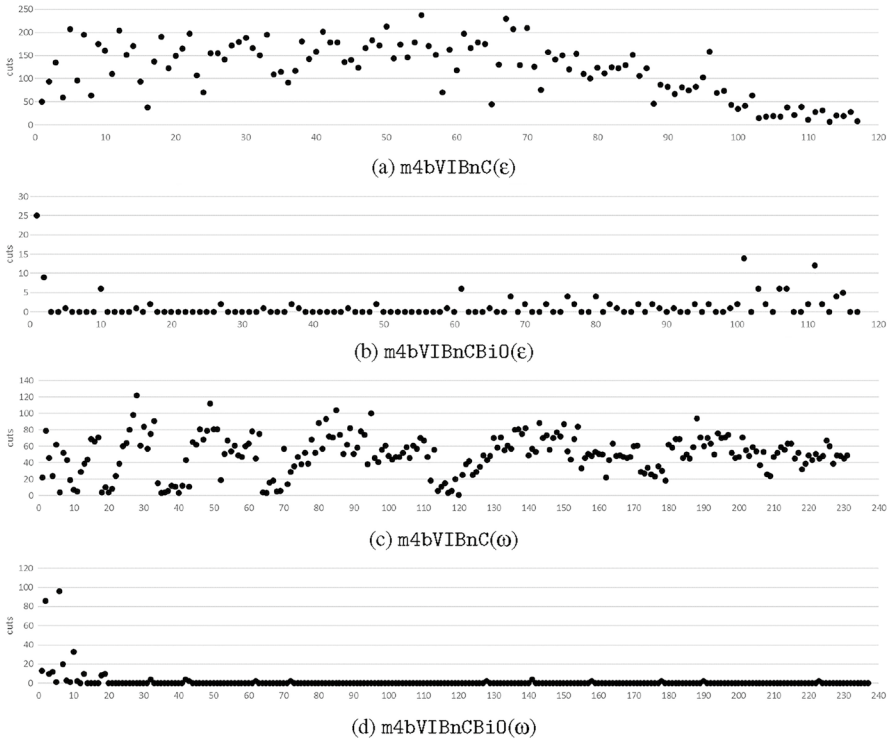


Fig. 2 Number of cuts added at each iteration for instance E_20_0. `m4bVIBnC`(ϵ) solves model `m4b` by branch-and-cut embedded in the ϵ-constraint method, and `m4bVIBnCBiO`(ϵ) additionally stores the detected infeasible paths to the set of constraints. `m4bVIBnC`(ω) solves model `m4b` by branch-and-cut and the weighting binary search method, and `m4bVIBnCBiO`(ω) additionally passes infeasible path constraints to subsequent iterations

Table 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.

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 different MOT compositions when solving different variants of the model. Note that the models provide the decision maker with a range of trade-off solutions. Based on this

Table 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))

$\|P\| = 20$	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	1694.9
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	–

solution pool, the decision maker derives actions and takes the best solution fitting to their requirements.

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

Figure 3 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 and, 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 comes from the additional time needed during specific day-times. Note that we usually have a $\beta > 1$, meaning that

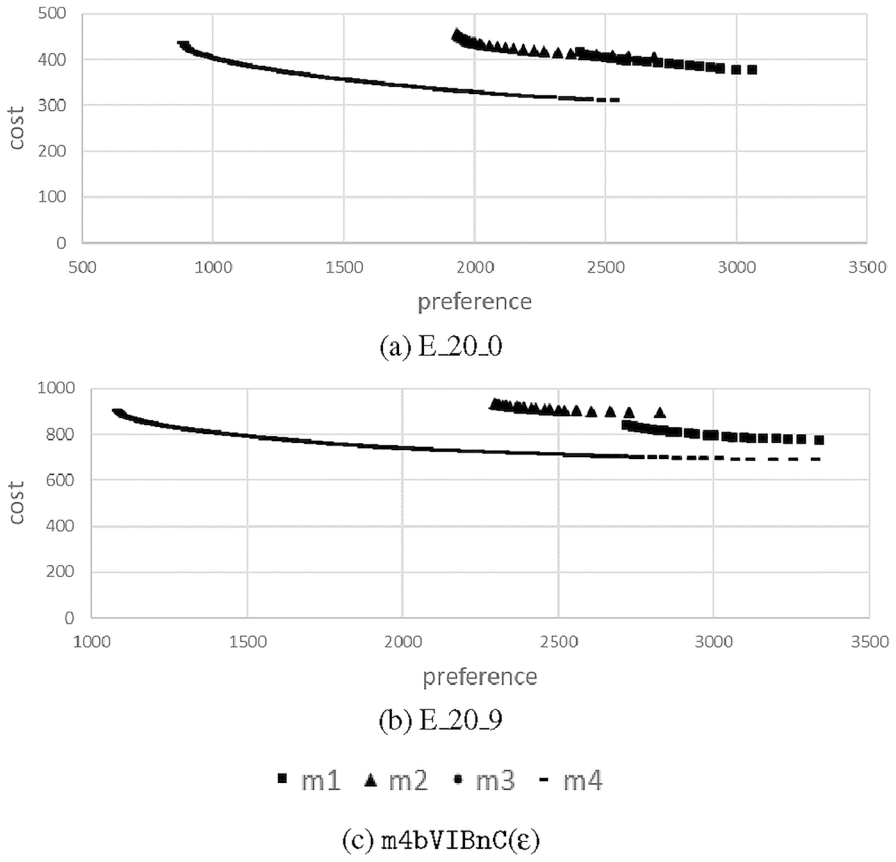


Fig. 3 Pareto frontiers for models `m1`, `m2`, `m3`, `m4` solving instances E_20_0 and E_20_9. The y-axis represents cost, preferences are on the x-axis

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 are obtained.

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

Finally, let us have a closer look at the MOTs assigned. We analyse 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 Fig. 4, 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 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 is 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)

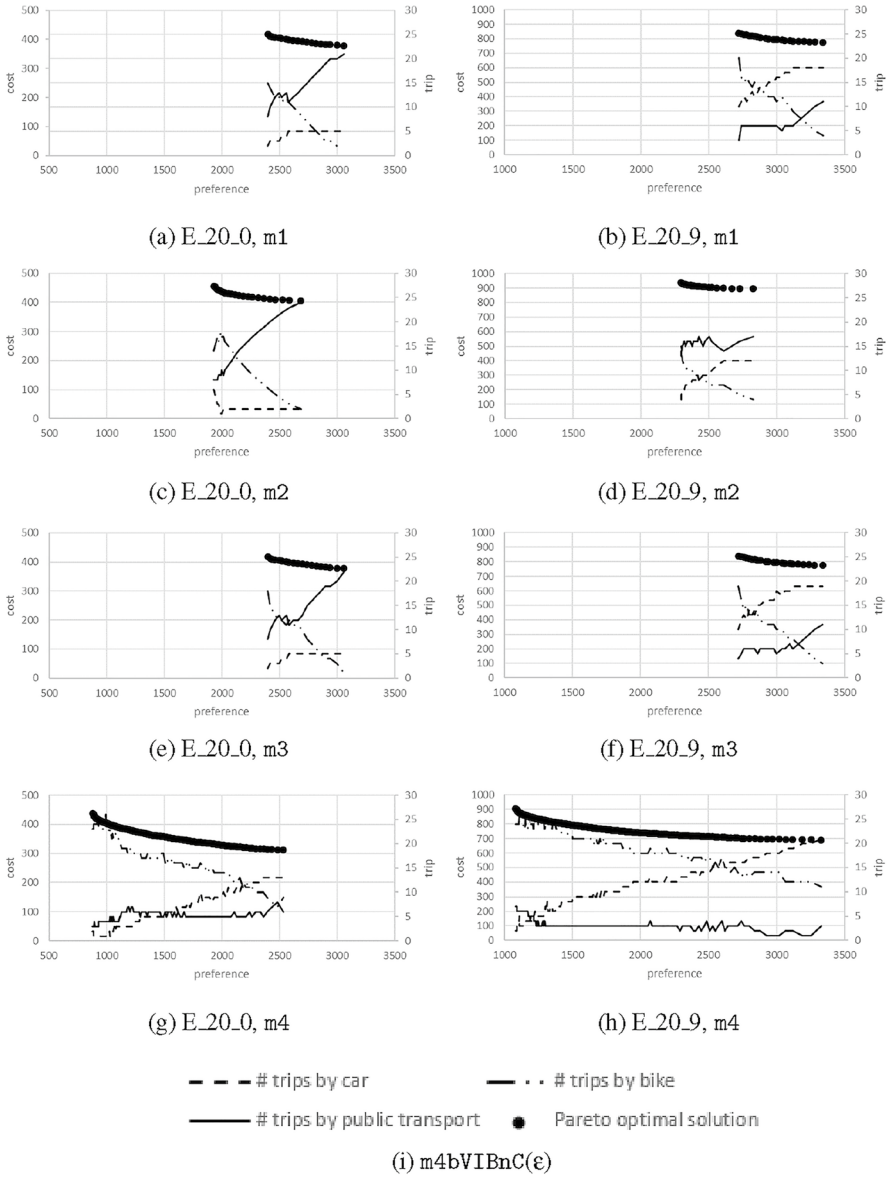


Fig. 4 Number of trips assigned for each Pareto optimal solution by the respective mode of transport (car, bike, public transportation) for models m1, m2, m3, m4 solving instances E_20_0 and E_20_9

preferences lead to more cars assigned and, conversely, more bikes are assigned with an increase in cost, and decrease in preferences.

Table 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

Table 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

	$\|P\|$	m1			m2			m3			m4		
		Car	Bike	Public	Car	Bike	Public	Car	Bike	Public	Car	Bike	Public
av	20	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		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	50	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		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	100	47.1	54.2	45.3	5.9	54.2	86.5	47.3	54.0	45.4	–	–	–
min		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	150	72.7	82.7	62.2	10.8	81.4	125.4	66.4	86.9	64.8	–	–	–
min		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	200	94.6	108.6	83.9	15.4	107.4	164.3	–	–	–	–	–	–
min		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	250	120.0	137.0	101.2	20.9	137.1	200.3	–	–	–	–	–	–
min		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	300	131.2	163.8	132.5	19.6	163.2	244.7	–	–	–	–	–	–
min		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	–	–	–	–	–	–

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 organized 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.

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 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 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.

Appendix

Algorithm 1: Separation of user routes.

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 & $\bar{x}_l = 1$) 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 4.2.4);
      end
    end
  end
end

Algorithm 2: Separation of car 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 4.2.4);
                    end
                end
            end
        end
    end
end
```

Table 10 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 11 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 10 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	Car	Walk/public	Bike
0	−3	−2	+1	1.2	1.1	1
1	+2	−2	+3	1.4	0.8	1.1
2	−2	−1	−3	1.3	0.9	1
3	+ 0	+ 0	+ 0	1	1	1
4	−2	−3	+1	1 .3	1	1
5	+2	−2	+3	1.4	0.9	1.1
6	−1	+2	−2	1.1	1.3	1

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.

Funding Open access funding provided by Austrian Science Fund (FWF).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Abad HKE, Vahdani B, Sharifi M, Etebari F (2018) A bi-objective model for pickup and delivery pollution-routing problem with integration and consolidation shipments in cross-docking system. J Clean Prod 193:784–801. https://doi.org/10.1016/j.jclepro.2018.05.046

Adelgren N, Gupte A (2017) Branch-and-bound for biobjective mixed integer programming. https://arxiv.org/abs/1709.03668

Al Maghraoui O, Vallet F, Puchinger J, Yannou B (2019) Modeling traveler experience for designing urban mobility systems. Des Sci 5:r7. https://doi.org/10.1017/dsj.2019.6

Alexiou D, Katsavounis S (2015) A multi-objective transportation routing problem. Oper Res Int Journal 15(2):199–211. https://doi.org/10.1007/s12351-015-0173-1

Anderluh A, Nolz PC, Hemmelmayr VC, Crainic TG (2019) Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone' customers arising in urban logistics. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2019.07.049

Androutsopoulos KN, Zografos KG (2017) An integrated modelling approach for the bicriterion vehicle routing and scheduling problem with environmental considerations. Transp Res Part C Emerg Technol 82:180–209. https://doi.org/10.1016/j.trc.2017.06.013

Aneja YP, Nair KPK (1979) Bicriteria transportation problem. Manag Sci 25(1):73–78

Ascheuer N, Fischetti M, Grötschel M (2000) A polyhedral study of the asymmetric traveling salesman problem with time windows. Networks 36(2):69–79. https://doi.org/10.1002/1097-0037(200009)36:2<69::AID-NET1>3.0.CO;2-Q

Ben Ticha H, Absi N, Feillet D, Quilliot A (2017) Empirical analysis for the VRPTW with a multigraph representation for the road network. Comput Oper Res 88:103–116

Ben Ticha H, Absi N, Feillet D, Quilliot A (2018) Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows. Comput Oper Res. https://doi.org/10.1016/j.cor.2018.11.001

Ben Ticha H, Absi N, Feillet D, Quilliot A, van Woensel T (2019) A branch-and-price algorithm for the vehicle routing problem with time windows on a road network. Networks. https://doi.org/10.1002/net.21852

Bérubé JF, Gendreau M, Potvin JY (2009) An exact ϵ-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits. Eur J Oper Res 194(1):39–50

Boland N, Charkhgard H, Savelsbergh M (2015) A criterion space search algorithm for biobjective integer programming: The balanced box method. Inf J Comput 27(4):735–754. https://doi.org/10.1287/ijoc.2015.0657

Braekers K, Hartl RF, Parragh SN, Tricoire F (2016) A bi-objective home care scheduling problem: analyzing the trade-off between costs and client inconvenience. Eur J Oper Res 248(2):428–443. https://doi.org/10.1016/j.ejor.2015.07.028

Bundesverband CarSharing eV (2020) CarSharing in Deutschland. https://www.carsharing.de/sites/default/files/uploads/infografiken_jahresbericht_2020-02.jpg. Last accessed on 28 May 2020

Calvo RW, de Luigi F, Haastrup P, Maniezzo V (2004) A distributed geographic information system for the daily car pooling problem. Comput Oper Res 31(13):2263–2278

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

Cattaruzza D, Absi N, Feillet D (2016) Vehicle routing problems with multiple trips. 4OR 14(3):223–259:223–259. https://doi.org/10.1007/s10288-016-0306-2

Ciari F, Bock B, Balmer M (2014) Modeling station-based and free-floating carsharing demand: test case study for Berlin. Transp Res Rec 2416(1):37–47

CIVITAS 202 (2020) Eccentric. https://civitas.eu/eccentric. Last accessed on 27 May 2020

de Almeida Correia GH, Antunes AP (2012) Optimization approach to depot location and trip selection in one-way carsharing systems. Transp Res Part E Logist Transp Rev 48(1):233–247

Deb K, Agrawal S, Pratap A, 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 JJ, Schwefel HP (eds) Parallel problem solving from nature PPSN VI. Springer, Berlin, Heidelberg, pp 849–858

Demir E, Bektaş T, Laporte G (2014) The bi-objective pollution-routing problem. Eur J Oper Res 232(3):464–478. https://doi.org/10.1016/j.ejor.2013.08.002

Doppstadt C, Koberstein A, Vigo D (2016) The hybrid electric vehicle - traveling salesman problem. Eur J Oper Res 253(3):825–842. https://doi.org/10.1016/j.ejor.2016.03.006

d'Orey PM, Ferreira M (2014) Can ride-sharing become attractive? a case study of taxi-sharing employing a simulation modelling approach. IET Intel Transp Syst 9(2):210–220

Ehrgott M, Gandibleux X (2003) Multiobjective combinatorial optimization–theory, methodology, and applications. Springer, Boston, MA. https://doi.org/10.1007/0-306-48107-3_8

Ehrgott M, Ljubić I, Parragh SN (2017) Feature cluster: recent advances in exact methods for multi-objective optimisation. Eur J Oper Res 260(3):805–806. https://doi.org/10.1016/j.ejor.2017.02.004

Eksioglu B, Vural AV, Reisman A (2009) The vehicle routing problem: a taxonomic review. Comput Ind Eng 57(4):1472–1483. https://doi.org/10.1016/j.cie.2009.05.009

Enzi M, Parragh SN, Pisinger D (2020a) Modeling and solving a vehicle-sharing problem. https://arxiv.org/abs/2003.08207

Enzi M, Parragh SN, Pisinger D, Prandtstetter M (2020b) Modeling and solving the multimodal car- and ride-sharing problem. https://arxiv.org/abs/2001.05490

Eskandarpour M, Ouelhadj D, Hatami S, Juan AA, Khosravi B (2019) Enhanced multi-directional local search for the bi-objective heterogeneous vehicle routing problem with multiple driving ranges. Eur J Oper Res 277(2):479–491. https://doi.org/10.1016/j.ejor.2019.02.048

Fahrrad W (2020) Radfahren in Zahlen. https://www.fahrradwien.at/radfahren-in-zahlen/radzahlen-2019/. Last accessed on 27 May 2020

Ferrero F, Perboli G, Rosano M, Vesco A (2018) Car-sharing services: an annotated review. Sustain Cities Soc 37:501–518. https://doi.org/10.1016/j.scs.2017.09.020

Garaix T, Artigues C, Feillet D, Josselin D (2010) Vehicle routing problems with alternative paths: an application to on-demand transportation. Eur J Oper Res 204(1):62–75. https://doi.org/10.1016/j.ejor.2009.10.002

Gendreau M, Ghiani G, Guerriero E (2015) Time-dependent routing problems: a review. Comput Oper Res 64:189–197. https://doi.org/10.1016/j.cor.2015.06.001

Ghannadpour SF, Zarrabi A (2019) Multi-objective heterogeneous vehicle routing and scheduling problem with energy minimizing. Swarm Evol Comput 44:728–747. https://doi.org/10.1016/j.swevo.2018.08.012

Gharari R, Poursalehi N, Abbasi M, Aghaie M (2016) Implementation of strength pareto evolutionary algorithm ii in the multiobjective burnable poison placement optimization of kwu pressurized water reactor. Nucl Eng Technol 48(5):1126–1139. https://doi.org/10.1016/j.net.2016.04.004

Govindan K, Jafarian A, Nourbakhsh V (2019) Designing a sustainable supply chain network integrated with vehicle routing: a comparison of hybrid swarm intelligence metaheuristics. Comput Oper Res 110:220–235. https://doi.org/10.1016/j.cor.2018.11.013
Grabenschweiger J, Tricoire F, Doerner KF (2018) Finding the trade-off between emissions and disturbance in an urban context. Flex Serv Manuf J 30(3):554–591. https://doi.org/10.1007/s10696-017-9297-3
Huang Y, Zhao L, van Woensel T, Gross JP (2017) Time-dependent vehicle routing problem with path flexibility. Transp Res Part B Methodol 95:169–175. https://doi.org/10.1016/j.trb.2016.10.013
Jozefowiez N, Semet F, Talbi EG (2008) Multi-objective vehicle routing problems. Eur J Oper Res 189(2):293–309. https://doi.org/10.1016/j.ejor.2007.05.055
MA 18 - Stadtentwicklung und Stadtplanung Wien (2015) Carsharing wien evaluierung. https://www.wien.gv.at/stadtentwicklung/studien/pdf/b008470.pdf. Last accessed on 28 May 2020
MA 46 (2020) Entwicklung des Radverkehrsnetzes in Wien (2000-2019). https://www.wien.gv.at/verkehr/radfahren/pdf/fakten-1.pdf. Last accessed on 27 May 2020
Maciejewski M, Salanova JM, Bischoff J, Estrada M (2016) Large-scale microscopic simulation of taxi services. Berlin and barcelona case studies. J Ambient Intell Humaniz Comput 7(3):385–393
Martin K (2020) Pkw-bestand in wien bis 2019. https://de.statista.com/statistik/daten/studie/683923/umfrage/pkw-bestand-in-wien/. Last accessed on 27 May 2020
Martínez-Salazar I, Ángel Bello F, Alvarez A (2015) A customer-centric routing problem with multiple trips of a single vehicle. J Oper Res Soc. https://doi.org/10.1057/jors.2014.92
Matl P, Hartl RF, 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. https://doi.org/10.1002/net.21876
Mourad A, Puchinger J, Chu C (2019) A survey of models and algorithms for optimizing shared mobility. Transp Res Part B Methodol 123:323–346. https://doi.org/10.1016/j.trb.2019.02.003
Nair R, Miller-Hooks E (2014) Equilibrium network design of shared-vehicle systems. Eur J Oper Res 235(1):47–61
Ngueveu SU, Prins C, Calvo RW (2010) A hybrid tabu search for the m-peripatetic vehicle routing problem. Springer, Boston, MA, pp 253–266. https://doi.org/10.1007/978-1-4419-1306-7_11
Nolz PC, Absi N, Feillet D (2014) A bi-objective inventory routing problem for sustainable waste management under uncertainty. J Multi-Criteria Decis Anal 21(5–6):299–314. https://doi.org/10.1002/mcda.1519
Paris en Selle (2020) Plan vélo. https://planvelo.paris/. Last accessed on 27 May 2020
Parragh SN, Tricoire F (2019) Branch-and-bound for bi-objective integer programming. Inf J Comput 31(4):805–822. https://doi.org/10.1287/ijoc.2018.0856
Riera-Ledesma J, Salazar-González JJ (2005) The biobjective travelling purchaser problem. Eur J Oper Res 160(3):599–613. https://doi.org/10.1016/j.ejor.2003.10.003
SEAMLESS (2020) SEAMLESS - sustainable, efficient austrian mobility with low-emission shared systems. http://www.seamless-project.at/projekt/. Last accessed on 02 May 2020
Silva MM, Subramanian A, Vidal T, Ochi LS (2012) A simple and effective metaheuristic for the minimum latency problem. Eur J Oper Res 221(3):513–520. https://doi.org/10.1016/j.ejor.2012.03.044
Sioui L, Morency C, Trépanier M (2013) How carsharing affects the travel behavior of households: a case study of montréal, canada. Int J Sustain Transp 7(1):52–69
Srinivasan V, Thompson GL (1976) Algorithms for minimizing total cost, bottleneck time and bottleneck shipment in transportation problems. Naval Res Logists Q 23(4):567–595. https://doi.org/10.1002/nav.3800230402
Stadtentwicklung Wien (2020) STEP2025 stadtentwicklungsplan wien. https://www.wien.gv.at/stadtentwicklung/studien/pdf/b008379a.pdf. Last accessed on 27 May 2020
Statistik W (2020) Wiener Bevölkerungsstand. https://www.wien.gv.at/statistik/bevoelkerung/bevoelkerungsstand/index.html. Last accessed on 27 May 2020
Stidsen T, Andersen KA, Dammann B (2014) A branch and bound algorithm for a class of biobjective mixed integer programs. Manag Sci 60(4):1009–1032. https://doi.org/10.1287/mnsc.2013.1802
Tachet R, Sagarra O, Santi P, Resta G, Szell M, Strogatz S, Ratti C (2017) Scaling law of urban ride sharing. Sci Rep 7:42868
Toro EM, Franco JF, Echeverri MG, Guimarães FG (2017) A multi-objective model for the green capacitated location-routing problem considering environmental impact. Comput Ind Eng 110:114–125. https://doi.org/10.1016/j.cie.2017.05.013

Toth P, Vigo D (2002) The vehicle routing problem. Society for Industrial and Applied Mathematics. doi:10(1137/1):9780898718515

Tricoire F, Parragh SN (2017) Investing in logistics facilities today to reduce routing emissions tomorrow. Transp Res Part B Methodol 103:56–67 (**green Urban Transportation**)

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 28 May 2020

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 27 May 2020

Vidal T, Laporte G, Matl P (2019) A concise guide to existing and emerging vehicle routing problem variants. CoRR abs/1906.06750, http://arxiv.org/abs/1906.06750

Vincent T, Seipp F, Ruzika S, Przybylski A, Gandibleux X (2013) Multiple objective branch and bound for mixed 0–1 linear programming: corrections and improvements for the biobjective case. Comput Oper Res 40(1):498–509. https://doi.org/10.1016/j.cor.2012.08.003

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

Weikl S, Bogenberger K (2013) Relocation strategies and algorithms for free-floating car sharing systems. IEEE Intell Transp Syst Mag 5(4):100–111

Wiener L (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 27 May 2020

Wiener L (2019) Modal split 2019. https://www.wien.gv.at/presse/bilder/2020/02/12/modal-split-2019_wiener-linien-png. Last accessed on 27 May 2020

Yh YV, Lasdon LS, Da Wismer D (1971) On a bicriterion formation of the problems of integrated system identification and system optimization. IEEE Trans Syst Man Cybern 3:296–297

Zhou B, Kockelman KM (2011) Opportunities for and impacts of carsharing: a survey of the austin, texas market. Int J Sustain Transp 5(3):135–152