[EURO Journal on Computational Optimization 9 (2021) 100010](https://doi.org/10.1016/j.ejco.2021.100010)

Contents lists available at [ScienceDirect](http://www.ScienceDirect.com/)

EURO Journal on Computational Optimization

journal homepage: [www.elsevier.com/locate/ejco](http://www.elsevier.com/locate/ejco)

Mobility offer allocations in corporate settings[☆](#_bookmark1)

Sebastian Knopp, Benjamin Biesinger[∗](#_bookmark2), Matthias Prandtstetter

*AIT Austrian Institute of Technology, Center for Energy – Integrated Energy Systems, Giefinggasse 4, Vienna 1210, Austria*

a r t i c l e i n f o a b s t r a c t

*2010 MSC:*

90C10

90C27

90C59

90B06

*Keywords:*

Mobility as a service Resource allocation

Integer linear programming Adaptive large neighborhood search

Corporate mobility is often based on a fixed assignment of vehicles to employees. Relaxing this fixation and including alternatives such as public transportation or taxis for business and private trips could increase fleet uti- lization and foster the use of battery electric vehicles. We introduce the *mobility oﬀer allocation problem* as the core concept of a flexible booking system for corporate mobility. The problem is equivalent to interval scheduling on dedicated unrelated parallel machines. We show that the problem is NP-hard to approximate within any factor. We describe problem specific conflict graphs for representing and exploring the structure of feasible solutions. A characterization of all maximum cliques in these conflict graphs reveals symmetries which allow to formulate stronger integer linear programming models. We also present an adaptive large neighborhood search based ap- proach which makes use of conflict graphs as well. In a computational study, the approaches are evaluated. It was found that greedy heuristics perform best if very tight run-time requirements are given, a solver for the integer linear programming model performs best on small and medium instances, and the adaptive large neighborhood search performs best on large instances.

# Introduction

The transportation sector is facing major changes due to digitaliza- tion and urbanization. Novel sharing concepts that offer mobility as a service arise. While such changes can be observed in many areas, cor- porate mobility has not changed significantly for decades. Many com- panies assign cars to certain employees based on their hierarchy level. Usually, such cars can be used for business and private trips. This fixed assignment of one car to one employee can result in rather large but ineﬃcient fleets with cars being used only less than one hour per day on average ([Bates and Leibling, 2012; Shoup, 2017](#_bookmark56)). Furthermore, such cars are often larger than necessary since all mobility needs of the em- ployee have to be covered with just one car. At the same time, other employees are neglected in such concepts. Often, an additional fleet of vehicles is available for business trips. These pool cars are booked using the first-come, first-served principle.

This paper proposes a novel concept for a corporate mobility ser- vice to improve this situation. The main idea is to focus on employees’ mobility demands rather than on fixed assignments of cars to employ- ees. Similar to car rental systems ([Oliveira et al., 2017](#_bookmark70)), travelers book mobility rather than a specific car.

A *mobility demand* can represent business travels (e.g. travel to a cus- tomer location for a meeting from 2:00pm to 4:00pm) as well as private

travels (e.g., weekend trip). A *mobility oﬀer* is one possible way to satisfy a mobility demand. For the meeting example from above, one mobility offer is to reserve a pool car from 1:30pm to 4:30pm. Another mobility offer is to take the train leaving at 1:15pm and returning at 4:45pm. The final assignment of cars or other transportation modes to travelers is done via an automatic mobility offer matching system.

This is modeled by the Mobility Offer Allocation Problem (MOAP). The corresponding optimization model and solution techniques are pre- sented in this paper. The objective is to determine an integrated alloca- tion that fulfills all mobility demands and respects the vehicle fleet size while minimizing expenses and emissions.

* 1. *A corporate mobility oﬀer concept with flexible bookings*

The Mobility Offer Allocation Problem was put into practice as part of an applied research project in Austria[1](#_bookmark0). This paper focuses on mod- eling and solving the Mobility Offer Allocation Problem. The research project addressed the corporate mobility concept in a broader way. The concept aims at improving costs and environmental impacts by shrink- ing the fleet size due to improved utilization, and by electrifying the

1 Project SEAMLESS (Sustainable Eﬃcient Austrian Mobility with Low- Emission Shared Systems), 2016–2019, <http://www.seamless-project.at/>, strategic research program “Leuchttürme der Elektromobilität”

☆ This work has been partially funded by the [Climate and Energy Fund](https://doi.org/10.13039/100008559)s (KliEn) within the strategic research program “Leuchttürme der Elektromobilität” under grant number [853767](#_bookmark1) (SEAMLESS).

∗ Corresponding author.

*E-mail addresses:* [Sebastian.Knopp@untis.at](mailto:Sebastian.Knopp@untis.at) (S. Knopp), [bbiesinger@gmail.com](mailto:bbiesinger@gmail.com) (B. Biesinger), [Matthias.Prandtstetter@ait.ac.at](mailto:Matthias.Prandtstetter@ait.ac.at) (M. Prandtstetter).

<https://doi.org/10.1016/j.ejco.2021.100010>

Received 23 November 2020; Received in revised form 22 August 2021; Accepted 25 August 2021

2192-4406/© 2021 The Author(s). Published by Elsevier Ltd on behalf of Association of European Operational Research Societies (EURO). This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>)

fleet due to increased flexibility (e.g., small electric cars for city trips, large combustion engine cars for holiday trips). To fully exploit those benefits, corporate mobility should be available for all employees and for business as well as for private trips. The following five main compo- nents were addressed in the implementation of this corporate mobility concept:

**Fleet of pool cars:** The appropriate mix and size of the fleet has to be determined based on the company’s specific needs. This fleet might consist of various car types: from small one- or two-seated city cars over vans to small- or medium-sized (light) trucks. Con- sidering private trips in addition to business trips could improve company fleet utilization while reducing the need for personally owned cars. In the literature, a lot of work is dedicated to find- ing optimal fleet compositions ([Hoff et al., 2010](#_bookmark92)). Battery elec- [tric cars should be incorporated to reduce emissions (Ma et al., 2012). Additionally, fleets could be made available not only to](#_bookmark60) employees but also to the public since this demand is often com- plementary.

**Integration of alternative mobility offers:** The concept of mobility [as a service should not be limited to fleet vehicles (Goodall et al., 2017). Alternative modes of transportation such as public trans-](#_bookmark81) portation, bicycle and car sharing systems, or taxis should be con- sidered as well. The system should automatically determine pos- sible alternative mobility offers. A seamless integration of such offers could advance the use of low-emission transport modes.

**Flexible booking and accounting system:** Users should be able to state preferences in an easy-to-use booking system. Reserving specific cars should be allowed only if necessary. Suitable mobil- ity offers are then assigned automatically. Additionally, the book- ing system needs to incorporate an accounting system where indi- vidual trips can be billed according to their context (e.g., business trips are accounted to the company while private trips are billed to the traveler).

**Key-less access:** A key-less solution eases access to cars. This means each user of the system should have a digital device to access and drive company cars. This could be an RFID chip card or an application on a mobile phone.

**Motivational strategies:** Appropriate motivational strategies should be applied in order to achieve the needed mind shift. Studies have shown that users are hardly willing to waive amenities they are already used to [Götz et al. (2011)](#_bookmark83). Economic incentives and motivational strategies as proposed in behavioral economics could be applied ([Metcalfe and Dolan, 2012](#_bookmark64)). Not only future travelers need to be convinced, but also upper man- agement, the accounting department, or the fleet management division.

* 1. *Related work*

From an application point of view, only few similar approaches have been found in the literature. A related system for sharing electrical vehicles in corporate contexts developed in an applied project is pre- sented in [Ostermann et al. (2014)](#_bookmark74) which underlines the relevance of the subject at hand. The method for scheduling vehicles within that system is described in [Koetter et al. (2015)](#_bookmark49). The proposed approach aims at minimizing fragmentation within the usage of the vehicle fleet in order to leave room for assigning novel requests. Their solution ap- proaches stem from control theory and operating systems, whereas our paper considers the problem from an operations research point of view. [Betz et al. (2016)](#_bookmark58) propose an approach which integrates the schedul- ing of charging battery electric vehicles. They propose a time indexed mixed integer linear programming (ILP) formulation with discrete time periods of 15 minutes. In this approach, each vehicle class is solved inde- pendently. The authors report computational times of 2 h for instances with 30 demands (called trips in their paper). A similar problem, also

including charging scheduling, is tackled in [Sassi and Oulamara (2017)](#_bookmark87). They propose a mixed integer linear programming model and a heuris- tic approach. For the exact approach, the authors report computational times of one hour for instances with about 120 demands (called tours in their paper). Their ILP model avoids overlapping vehicle assignments by including one constraint for each possible vehicle assignment. We use a stronger formulation based on cliques in a conflict graph. Though not explicitly considering the charging process of vehicles (and thus being less general regarding this aspect), the problem considered in our paper is more general with regard to mobility options. In particular, vehicle dependent journey intervals (possibly mode of transport dependent) and alternative journey intervals (e.g., modeling appointment alternatives) are considered, whereas the approaches from the literature mentioned before assume identical journey intervals for each vehicle. Also, our ap- proach can include different vehicle admissibility for each demand.

In this paper, the problem is modeled as a generalized operational fixed interval scheduling problem. Consider [Kolen et al. (2007)](#_bookmark51) for a sur- vey on interval scheduling and [Kovalyov et al. (2007)](#_bookmark53) for fixed interval scheduling. [Kovalyov et al. (2007)](#_bookmark53) define the fixed interval scheduling

problem as follows. We are given *𝑛* independent non-preemptive jobs to

be processed on *𝑚* independent parallel machines, where each machine

can process at most one job at a time. Each job has a machine-dependent

weight and a set of fixed intervals in which it can be processed at a specific machine. Now, the aim of the *tactical* fixed interval scheduling problem is to minimize the number of needed machines given that all jobs have to be scheduled ([Kroon et al., 1997](#_bookmark54)). The goal of the *opera- tional* fixed interval scheduling problem is to maximize the weight of [the jobs that can be scheduled on a given set of machines (Ng et al., 2014). In our application, machines correspond to vehicles, jobs to mo-](#_bookmark66) bility demands, and the possible assignments of a job to a machine at a specific interval to mobility offers. [Ng et al. (2014)](#_bookmark66) present heuristics for a special case of the operational fixed interval scheduling problem, to which we compare our developed algorithms in [Section 6](#_bookmark24).

[Similarly, *vehicle scheduling* problems (see Bunte and Kliewer (2009) for an overview) are often found in the context of public](#_bookmark61) transport planning. They deal with the task of assigning and sequenc- ing vehicles to trips with fixed travel times. Some variants of vehicle scheduling account for multiple vehicle types ([Hassold and Ceder, 2014](#_bookmark91)) [or allow slightly changing the timetable of the trips (Desfontaines and Desaulniers, 2018). Classical vehicle scheduling problems deal with](#_bookmark67) public transport, e.g., bus service planning. Another variant is the *rental vehicle scheduling* or *vehicle-reservation assignment* problem. In these problems, there are usually multiple depots where the fleet is located, there are dynamic booking requests, substitutions (e.g., to a better vehicle class) are possible, and there is the need of vehicle relocations between the depots. [Oliveira et al. (2017)](#_bookmark70) give an overview of problems arising in the context of fleet and revenue management of car rental companies. In [Ernst et al. (2011)](#_bookmark71), a car rental problem is modeled using a set of assignment problems with linking constraints and tackled using a Lagrangean heuristic. A real-world use case of such problems is shown in [Ernst et al. (2007)](#_bookmark72) in which a fleet of around 4000 vehicles of a company located in Australia and New Zealand is scheduled. Another practical application was tackled in [Oliveira et al. (2014)](#_bookmark73) which also presents an integer linear programming model and a matheuristic to solve the problem.

Interval scheduling problems are based on the concept of interval graphs, in which conflicts between intervals are represented as undi- rected edges. Interval graphs are a widely studied graph class in algorith- mic graph theory, e.g., as a subclass of chordal graphs ([Farber, 1984](#_bookmark76)). Since in our case each interval has a transport mode dependent cost, the problem is also close to the weighted interval graph coloring problem which is proven to be NP-hard in [Escoﬃer et al. (2005)](#_bookmark75). Another related problem is the maximum weighted independent set problem for interval graphs which is discussed, e.g., in [Bar-Noy et al. (2001)](#_bookmark52). In contrast to the problem considered in our work, the maximum weighted indepen- dent set problem is, however, solvable in polynomial-time because of

its restriction to interval graphs. Another related problem is the interval scheduling problem with a resource constraint ([Angelelli et al., 2014](#_bookmark48)) which is, in contrast to the MOAP, not a variant of a multiple-interval scheduling problem. Many problems, like independent set, dominating set, and clique, are shown to be NP-hard for multiple-interval graphs whereas they are not for 1-interval graphs ([Butman et al., 2010](#_bookmark65)). Similar models are also used in course timetabling, consider, e.g., the overview provided in [Burke et al. (2010)](#_bookmark62). Related are also variants of graph col- oring problems ([Marx, 2004](#_bookmark63)).

* 1. *Contributions and structure of this paper*

We propose a flexible booking system which leads to the introduction of an NP-hard *mobility oﬀer allocation problem*. Solution methods with varying trade-offs between run-time and solution quality are described and evaluated. This work complements the existing literature as it fo- cuses on a real-world application of multi-interval scheduling which has found only little attention in the literature. The mobility offer allocation problem is equivalent to interval scheduling on dedicated unrelated par- allel machines ([Ng et al., 2014](#_bookmark66)). To emphasize the application specific focus of this paper, we use the name *mobility oﬀer allocation problem*. We will demonstrate its relatation to fixed interval scheduling problems and show that MOAP is NP-hard to approximate. Known heuristics are out- performed by the proposed methods. We define problem specific conflict graphs for representing and exploring the structure of feasible solutions. We develop a characterization of all maximum cliques in these conflict graphs, revealing symmetries which allow to formulate stronger integer linear programming models. We also present an adaptive large neigh- borhood search based approach which makes use of conflict graphs as well. A computational study using two sets of benchmark instances con- firms that, as one would expect, the greedy heuristics perform best if very tight run-time requirements are given, a solver for the integer lin- ear programming model performs best on small and medium instances, and the adaptive large neighborhood search performs best on large in- stances. The integer linear programming approach of this paper solves instances with up to 200 demands in less than one second. Instances with up to 2000 demands are solved to optimality within one hour of computational time.

The outline of this paper is as follows. [Section 2](#_bookmark3) formally defines the

problem and discusses the modeling. [Section 3](#_bookmark6) defines conflict graphs as a foundation for the solution approaches proposed in [Section 4](#_bookmark9). Then, [Section 5](#_bookmark21) discusses how symmetries can be exploited by grouping ve- hicles into disjoint classes. A description of benchmark instances along with a computational study is presented in [Section 6](#_bookmark24), followed by con- clusions and an outlook in [Section 7](#_bookmark45).

# Problem description and complexity

First, this section formally specifies the problem that is investigated. Then, it discusses how that model can capture various practical require- ments. Finally, the section shows that the problem is NP-hard to approx- imate within any factor.

* 1. *Formal problem description*

*bility demands 𝐷* and a fleet of *vehicles 𝑉* representing resources with In the *Mobility Oﬀer Allocation Problem*, we are given a set of *mo-* limited availability. For each mobility demand *𝑑* ∈ *𝐷*, we are given a

∗ if no vehicle is needed (e.g., if the mobility offer corresponds to using

public transport).

The problem is to *select* exactly one mobility offer for each demand such that the total cost of the selected offers is minimal while overall feasibility is ensured. Feasibility is achieved if for each pair of selected

offers *𝑜*1 *, 𝑜*2 ∈ *𝑂* with *𝑣𝑜*1 = *𝑣𝑜*2 ≠∗ it holds that *𝑇𝑜*1 ∩ *𝑇𝑜*2 = ∅, i.e., the

journey intervals of all selected offers that use the same vehicle do not overlap.

* 1. *Discussion*

Although the problem description given above might appear simplis- tic, many features and aspects of practical concerns can be included due to the way mobility offers are generated. We consider this simplicity to be an important advantage of the chosen modeling. A mobility of- fer can comprise a complex itinerary consisting of many different loca- tions. In a real world setting, the offers are not given beforehand, instead they are computed on demand. Known approaches for route planning (see [Bast et al. (2016)](#_bookmark55) for an overview) can be employed for computing routes for the given vehicles and user needs to accurately compute jour- ney intervals. In particular, alternative modes of transports in addition to the given fleet of cars can be included. This can, e.g., comprise public transport operators, taxi cooperations, and bike sharing providers. We assume an infinite capacity for these modes. Related mobility offers do

not require a vehicle from the given fleet (i.e., *𝑣𝑜* =∗) and thus are never

in conflict to each other. Note that the journey intervals of offers not only

include travel times, but also service times, waiting times, or even visits of multiple customers with additional travel times in between. Different offers belonging to the same demand may feature differing journey in- tervals. In particular, it is possible to include multiple offers per vehicle with different journey intervals for modeling alternative dates for the same appointment. Demands can also be used to model maintenance tasks for the vehicles of the fleet, with corresponding offers defining possible maintenance dates.

Multiple offers can require the same vehicle at the same time. This

demands must be disjoint. Consider for example two demands *𝐴* and *𝐵* does not violate the requirement that the offer sets of different mobility which both could be satisfied by using a vehicle *𝑣*. In this example, we have an offer *𝑜𝐴* whose selection would mean the car will be assigned to satisfy demand *𝐴*, and a different offer *𝑜𝐵* whose selection would mean the car will be assigned to satisfy demand *𝐵*. In this case, at most one

of these offers can be selected in a feasible solution.

A core assumption is that each vehicle is assigned to a fixed location where all trips start and end. Yet, different vehicles can be located at different places. This assumption holds in many corporate contexts and also in related use-cases such as, e.g., car sharing systems where each vehicle is bound to one fixed location. Beside the administrative rea- sons observable in practice (e.g., maintenance responsibilities), station based systems have some inherent advantages: First, there is no need to consider vehicle relocations due to the imbalance of travel demand. Sec- ond, station based systems are robust against canceled trips. Canceling

but canceling a trip from *𝐴* to *𝐵* in a more flexible system could ren- a trip in advance cannot cause a problem if vehicle locations are fixed, der other trips starting from *𝐵* impossible. Certainly, there are scenarios

where allowing relocations is reasonable. However, such scenarios are

not considered in this work.

set of *mobility oﬀers 𝑂𝑑*

forming the overall set of offers *𝑂* = ⋃*̇*

*𝑑*∈*𝐷*

*𝑂𝑑* .

Especially in practical applications, feasibility of the problem in-

Note that for different mobility demands *𝑑*1 ≠ *𝑑*2 ∈ *𝐷* we assume the sets *𝑂𝑑*1 *, 𝑂𝑑*2 to be disjoint. Each mobility offer *𝑜* ∈ *𝑂* is associated with a *cost 𝑐𝑜* ∈ ℝ and an *journey interval 𝑇𝑜* = [*𝑎𝑜 , 𝑏𝑜* ) with *𝑎𝑜 , 𝑏𝑜* ∈ ℝ defining its start time *𝑎𝑜* and end time *𝑏𝑜* (assuming *𝑎𝑜 < 𝑏𝑜* ). The *duration* of *𝑇𝑜* is denoted by *𝜏𝑜* = *𝑏𝑜* − *𝑎𝑜* . A mobility offer might require a *vehicle*, so

*𝑣𝑜* ∈ *𝑉* ∪ {∗} specifies for each offer *𝑜* ∈ *𝑂* either the required vehicle or

stances cannot always be ensured. A reasonable assumption, however, is to add an offer to each demand denoting a taxi trip which has high

offer *𝑜* ∈ *𝑂* representing a regret cost can be added to each demand de- cost but is always feasible. When taxis are not available, an artificial

noting that this demand cannot be fulfilled. Therefore, infeasibilities are not explicitly considered in the proposed solution algorithms.

* 1. *Complexity* identify mobility offers that use the same car and have overlapping

journey intervals. Each conflict graph *𝐺𝑣* is an interval graph. Sec-

*𝑂*

The mobility offer allocation problem is equivalent to the interval

ondly, we have for each mobility demand *𝑑* ∈ *𝐷* a conflict graph *𝐺𝑑* =

scheduling on dedicated unrelated parallel machines (ISDU) introduced

(*𝑂 , 𝐸* ) with *𝐸*

= {{*𝑜, 𝑜*′} ∣ *𝑜, 𝑜*′ ∈ *𝑂* }

*𝑂*

*𝐺𝑑*

*𝑑 𝑑 𝑑*

*𝑑* . Each conflict graph

*𝑂* is a

in [Ng et al. (2014)](#_bookmark66). A mapping between instances of ISDU and instances

of MOAP is given by identifying machines with vehicles, jobs with mo-

complete graph since exactly one offer must be chosen for each de-

mand. Then, the *oﬀer conflict graph 𝐺𝑂* = (*𝑂, 𝐸*) is defined by setting

bility demands, and intervals with mobility offers. Unavailability inter- vals can be represented by artificial mobility demands with a single as-

⋃

*𝑑*∈*𝐷*

*𝐸* =

∪

*𝐸𝑑*

⋃

*𝑣*∈*𝑉*

*𝐸𝑣*.

sociated offer. ISDU is shown to be NP-hard in [Ng et al. (2014)](#_bookmark66) by stating that it is a generalization of interval scheduling on dedicated identical [parallel machines (ISDI), which is shown to be NP-hard by Arkin and Sil- verberg (1987). Throughout this paper we use the notation introduced](#_bookmark50)

for the mobility offer allocation problem as this allows us to discuss

* 1. *Cliques in oﬀer conflict graphs*

In a general undirected graph *𝐺* = (*𝑉 , 𝐸*), a subset of nodes *𝐶 ⊆ 𝑉* is a *clique* if and only if there exists an edge between each pair of nodes

in *𝐶*, i.e., ∀ *𝑐 , 𝑐* ∈ *𝐶* ∶ { *, 𝑐* } ∈ *𝐸*. A clique *𝐶* is a *maximum clique* if *𝐺*

*𝑐*

1 2 1 2

in terms of the problem domain. In addition, [Theorem 1](#_bookmark4) relates MOAP to another problem from the literature and shows the stronger result that MOAP is NP-hard to approximate within any factor by reducing the interval scheduling problem with machine availabilities (ISMA) to the mobility offer allocation problem (MOAP). These results suggests that heuristics might be necessary for solving large and diﬃcult instances.

**Theorem 1.** *If 𝑃* ≠ *𝑁𝑃 , then for any factor there is no polynomial-time*

*approximation algorithm for the mobility oﬀer allocation problem.*

**Proof.** We show that the MOAP is NP-hard to approximate within any factor by a polynomial-time reduction from the Interval Scheduling with Machine Availabilities (ISMA) problem which has been shown to be NP-complete in [Kolen et al. (2007)](#_bookmark51). An instance of the ISMA problem

is defined as follows: There are *𝑚* machines, continuously available in

[*𝑎𝑖 , 𝑏𝑖* ] with *𝑖* = 1*,* … *, 𝑚* and *𝑛* jobs requiring processing from *𝑠𝑗* to *𝑓𝑗* with

*𝑗* = 1*,* … *, 𝑛*. The question is whether a feasible schedule exists such that

each job is processed by a machine within its availability interval such

by creating *𝑛* mobility demands and *𝑚* vehicles. For each mobility de- that no two jobs overlap. From that we construct an instance of MOAP mand *𝑑𝑗* with *𝑗* = 1*,* … *𝑛* there is a corresponding offer *𝑜* for each vehicle

does not contain another clique *𝐾*, such that *𝐶 ⊂ 𝐾*. For general graphs,

the number of maximum cliques can be exponential in the number of

nodes ([Moon and Moser, 1965](#_bookmark68)). However, all maximum cliques in the offer conflict graph can be enumerated eﬃciently and their number is

at most |*𝐷*| + ∑ |*𝑂* |. This is shown in the following.

*𝑣*∈*𝑉* | *𝑣* |

**Theorem 2.** *If 𝐾 ⊂ 𝑉 is a maximum clique in a mobility oﬀer conflict*

*graph, then*

*-* ∃ *𝑑* ∈ *𝐷 such that 𝐾* = *𝑂𝑑 , or*

*-* ∃ *𝑣* ∈ *𝑉 such that 𝐾 is a maximum clique in 𝐺𝑣 .*

*𝑂*

**Proof.** Let *𝐾 ⊂ 𝑉* be a maximum clique.

**Case 1.** If ∃ *𝑑* ∈ *𝐷* such that *𝐾 ⊆ 𝑂𝑑* , then *𝐾* = *𝑂𝑑* since *𝐺𝑑* = (*𝑂𝑑 , 𝐸𝑑* )

*𝑂*

is a complete graph.

**Case 2.** If ∄ *𝑑* ∈ *𝐷* such that *𝐾 ⊆ 𝑂𝑑* , then ∃ *𝑐* ≠ *𝑑* ∈ *𝐷* with *𝐾* ∩ *𝑂𝑐* ≠ ∅ and *𝐾* ∩ *𝑂𝑑* ≠ ∅. Further, there must exist an edge {*𝑎, 𝑏*} ∈ *𝐸* with *𝑎* ∈ *𝑂𝑐* and *𝑏* ∈ *𝑂𝑑* since *𝐾* is a clique. By construction of the conflict graph, this edge can be only induced by a vehicle conflict because *𝑐* and *𝑑*

belong to different demands. Thus, we have *𝑣𝑎* = *𝑣𝑏* . Now, denoting *𝑣* =

*𝑚𝑖* with *𝑖* = 1*,* … *, 𝑛* if [*𝑠𝑗 , 𝑓𝑗* ] *⊆* [*𝑎𝑖 , 𝑏𝑖* ] with *𝑐𝑜* = 0 and *𝑇𝑜* = [*𝑠𝑗 , 𝑓𝑗* ]. Ad-

ditionally, there is an offer *𝑜*′ for each demand *𝑑*1 *,* … *, 𝑑𝑛* with *𝑐𝑜*′ = 1

*𝑣𝑎* = *𝑣𝑏* , assume there is a node *𝑘* ∈ *𝐾* with *𝑣𝑘* ≠ *𝑣*. Since there is no

and *𝑣𝑜*′ =∗. Then, the result of the question whether there exists a fea-

sible allocation of exactly one offer to each demand with cost smaller

vehicle conflict, *𝑘* can be connected to both nodes, *𝑎* and *𝑏*, only due to

a demand conflict. However, there cannot be a demand conflict to both

nodes at the same time, since they correspond to different demands.

than 1 is also an answer to the ISMA problem. Now, assume there exists a polynomial-time algorithm for MOAP which is able to find a solu-

tion within a factor of *𝛼* to the optimal solution. In case there exists

a feasible schedule for the given ISMA problem the objective function

value found by that approximation algorithm must be zero. Thus, the

Since *𝐾* is a maximum clique, there cannot be a node *𝑘* ∈ *𝐾* with *𝑣𝑘*

thus *𝐾 ⊆ 𝑂𝑣* .

□

≠ *𝑣*,

ISMA problem. Unless *𝑃* ≠ *𝑁𝑃* , such an algorithm cannot exist. □ polynomial-time approximation algorithm for MOAP would solve the

# Conflict graphs

Conflict graphs are a well-known modeling technique, used, e.g., for solving coloring or scheduling problems. They are a funda- mental concept for the solution approaches proposed in this pa- per. In particular, interval graphs in interval scheduling as discussed in [Kolen et al. (2007)](#_bookmark51) are conflict graphs. An interval graph is an undi- rected graph whose nodes correspond to intervals on the real number line and whose edges identify overlaps between the intervals. Similarly, we define an offer conflict graph for identifying all possible conflicts between mobility offers. Nodes in the conflict graph correspond to mo- bility offers. Edges identify pairs of offers that may not be selected at the same time. Subsequently, based on the offer conflict graph, cliques in that offer conflict graph are identified and a demand conflict graph is introduced.

* 1. *Oﬀer conflict graphs*

First, for each vehicle *𝑣* ∈ *𝑉* , a conflict graph *𝐺𝑣* = (*𝑂𝑣, 𝐸𝑣* ) is de-

[Theorem 2](#_bookmark5) shows that there are only two types of maximum cliques in the mobility offer conflict graph. Next, we describe the construc- tion of the conflict graph by enumerating the cliques identified above. The first type of cliques (offers belonging to the same demand) can be derived directly from the problem instance. The second type of cliques (offers belonging to the same vehicle with overlapping time in- tervals) can be computed independently for each vehicle. As proposed in [Gupta et al. (1982)](#_bookmark88), this can be done by adapting the algorithm of [Gupta et al. (1979)](#_bookmark85) for finding a minimum coloring of an interval graph. The following algorithm describes that adaption.

in an interval graph *𝐺* = (*𝑉 , 𝐸*). We assume the interval graph to be **Algorithm 1.** This algorithm successively reports all maximum cliques

given by its implicit representation, i.e., a set of intervals in the real line. The start and end dates of the intervals are denoted as left and right endpoints, respectively.

1. Maintain an initially empty set of nodes *𝐶*, representing the current

maximum clique candidate.

1. Sort the 2 ⋅ |*𝑉* | endpoints of the intervals of *𝑉* in ascending order. In

case of ties with left and right endpoints of journey intervals, right

endpoints always come first.

1. Scan the list of sorted endpoints. Let *𝑒* be the current endpoint.

fined. Its nodes *𝑂𝑣* =

{ }

*𝑜* ∈ *𝑂* ∣ *𝑜𝑣* = *𝑣*

*𝑂*

correspond to all offers requiring

If *𝑒* is a left endpoint: Add the corresponding offer to *𝐶*.

vehicle *𝑣*. The edges of the graph *𝐸𝑣* = {{*𝑜, 𝑜*′} ∈ *𝑂*

*𝑣*

× *𝑂𝑣*

∣ *𝑇𝑜*

}

∩ *𝑇𝑜*′ ≠ ∅

If *𝑒* is a right endpoint:

* 1. If the previous endpoint was a left endpoint, report *𝐶* as a

newly found maximum clique.

* 1. Remove *𝑒* from the current maximum clique candidate *𝐶*. The runtime complexity of [Algorithm 1](#_bookmark7) is U(|*𝑉* | ⋅ log |*𝑉* |), not includ-

The number of maximum cliques in an interval graph *𝐺* = (*𝑉 , 𝐸*) is at ing the output complexity of reporting newly found maximum cliques. most |*𝑉* | since for each node in [Algorithm 1](#_bookmark7) at most one maximum clique is reported. Since each clique can contain at most |*𝑉* | nodes, the run-

maximum cliques is U(|*𝑉* |2). time complexity of the algorithm including the effort for reporting all

Now, we can make the following observation regarding the number of maximum cliques.

**Corollary 1.** *The number of maximum cliques in a mobility oﬀer conflict*

intervals between *𝐴*2 and *𝐶*1, *𝐴*3 and *𝐶*2, and *𝐶*2 and *𝐵*1. Feasible selec- tions, e.g., are {*𝐴*3*, 𝐵*1*, 𝐶*1} or {*𝐴*2*, 𝐵*2*, 𝐶*2}. The demand conflict graph

is indicated by background shapes which group nodes that correspond to the same demand. These three shapes form the nodes of the demand

conflict graph. For example, the set of nodes {*𝐴*1*, 𝐴*2*, 𝐴*3*, 𝐴*4} in the offer

conflict graph corresponds to the node representing demand *𝐴* in the de-

mand conflict graph. There are two edges in the demand conflict graph

edge connects the nodes corrsponding to the demand *𝐴* and *𝐶*, another which are indicated by the large dotted lines connecting the shapes. One connects the demands *𝐵* and *𝐶*.

# Solution approaches

We propose a variety of solution approaches including an ILP model

*graph is at most* |*𝐷*| + ∑

|*𝑂* |*.*

using a general purpose solver, greedy algorithms, and an adaptive large

*𝑣*∈*𝑉* | *𝑣* |

neighborhood search. All are evaluated against two sets of benchmark

maximum cliques in an interval graph *𝐺* = (*𝑉 , 𝐸*) is at most |*𝑉* |. □ **Proof.** This follows from [Theorem 2](#_bookmark5) and the fact that the number of

* 1. *Demand conflict graphs*

We now introduce demand conflict graphs in order to identify po- tentially conflicting mobility demands. These graphs are applied in [Section 4.3](#_bookmark17) for defining an eﬃcient destroy operator of a large neigh- borhood search based approach. The following definition of a demand conflict graph is described in terms of the quotient graph of an offer conflict graph. Beforehand, we recall the definition of a quotient graph which is a known concept from graph theory (see, e.g., [Gustin (1963)](#_bookmark89) or [Sanders and Schulz (2012)](#_bookmark86)).

A *quotient graph 𝐺𝑞* = (*𝑉𝑞 , 𝐸𝑞* ) is defined for a given partitioning

instances in [Section 6](#_bookmark24). For solving the ILP, the mixed integer linear pro- gramming solver IBM ILOG CPLEX Optimizer, version 12.6.2, is used. For larger instances, this exact approach is not successful anymore which is why we also propose heuristic solution algorithms. All proposed meth- ods make use of the conflict graphs introduced in [Section 3](#_bookmark6).

* 1. *Integer linear programming model*

The definition of the offer conflict graph leads to an integer linear programming model which is presented next. We define binary decisions

variables *𝑥𝑜* ∈ {0*,* 1} for each mobility offer *𝑜* ∈ *𝑂* denoting whether or

conflict graph *𝐺𝑣* of a vehicle *𝑣* ∈ *𝑉* by *𝐶 𝑣*. not an offer is selected. We denote the set of maximum cliques in the

∑

*𝑚𝑖𝑛 𝑥𝑜* ⋅ *𝑐𝑜* (1)

*𝑉* = ⋃*̇ 𝑘 𝑉* of the nodes of an original graph *𝐺* = (*𝑉 , 𝐸*). The nodes *𝑉*

*𝑜* ∈ *𝑂*

*𝑖*=1 *𝑖*

{ } *𝑞*

of the quotient graph are then given by *𝑉𝑞* = *𝑉*1*, 𝑉*2*,* … *, 𝑉𝑘* . There ex- ∑

{ }

ists an edge *𝑉𝑖 , 𝑉𝑗*

∈ *𝐸𝑞* between two nodes *𝑉𝑖 , 𝑉𝑗* ∈ *𝑉𝑞* in the quotient

*𝑠.𝑡.*

*𝑥𝑜* = 1 ∀ *𝑑* ∈ *𝐷* (2)

graph *𝐺𝑞*

if and only if there exists an edge {*𝑎, 𝑏*} ∈ *𝐸* with *𝑎* ∈ *𝑉𝑖*

and

*𝑜* ∈ *𝑂𝑑*

*𝑏* ∈ *𝑉𝑗* in the original graph *𝐺*. ∑

A *demand conflict graph 𝐺𝐷* = (*𝐷, 𝐸𝐷* ) is defined as the quotient graph

*𝑥𝑜* ≤ 1 ∀ *𝑣* ∈ *𝑉 ,* ∀ *𝐾* ∈ *𝐶 𝑣* (3)

of an offer conflict graph *𝐺𝑂* = (*𝑂, 𝐸*) with a partitioning of the offers

*𝑜* ∈ *𝐾*

*𝑂* = ⋃*̇*

*𝑑*∈*𝐷*

*𝑂𝑑*

given by the demands *𝑑* ∈ *𝐷*. So, each node in a *demand*

*conflict graph 𝐺𝐷* = (*𝐷, 𝐸𝐷* ) is associated to a mobility demand *𝑑* ∈ *𝐷*. The set of edges *𝐸𝐷* contains an edge {*𝑑, ℎ*} with *𝑑, ℎ* ∈ *𝐷* if there is a potential conflict between the corresponding demands *𝑑* ∈ *𝐷* and *ℎ* ∈ *𝐷*. A potential conflict between demands *𝑑* and *ℎ* exists if there is an offer *𝑎* ∈ *𝑂𝑑* conflicting with an offer *𝑏* ∈ *𝑂ℎ* , i.e., both offers have

overlapping journey intervals requiring the same vehicle. Note that de-

mand conflict graphs are related to multiple-interval graphs as described in [Butman et al. (2010)](#_bookmark65). The perspective of demand conflict graphs also allows to see the MOAP as a weighted variant of the known list col- oring problem described in [Marx (2004)](#_bookmark63), with colors corresponding to vehicles.

* 1. *Example*

[Fig. 1](#_bookmark16) shows an example of a MOAP instance with two vehicles *𝑉*1 and *𝑉*2. It includes three mobility demands *𝐴*, *𝐵*, and *𝐶* with mobility of- fers {*𝐴*1*, 𝐴*2*, 𝐴*3*, 𝐴*4}, {*𝐵*1*, 𝐵*2}, and {*𝐶* 1*, 𝐶* 2*, 𝐶* 3}, respectively. The jour- ney intervals of these offers are shown in [Fig. 1](#_bookmark16)a. Note that demand *𝐴* demonstrates alternative appointment dates since the offers *𝐴*1 and *𝐴*2

require the same vehicle during different journey intervals. The corre- sponding conflict graphs are shown in [Fig. 1](#_bookmark16)b. Circles represent nodes in the offer conflict graph. Conflict edges between two offers of the same demand are drawn using dashed lines. Conflicts related to vehicles with overlapping journey intervals are drawn using straight lines. As indi-

*𝑥𝑜* ∈ {0*,* 1} ∀ *𝑜* ∈ *𝑂* (4)

The objective function [(1)](#_bookmark11) minimizes the sum of the costs of selected offers. Constraints [(2)](#_bookmark13) ensure that for each mobility demand exactly one offer is selected. Constraints [(3)](#_bookmark14) prevent selecting offers which require the same vehicle at the same time. Constraints [(2)](#_bookmark13) and [(3)](#_bookmark14) directly cor- respond to the two types of maximum cliques identified in [Theorem 2](#_bookmark5) of [Section 3.1](#_bookmark8).

of the form *𝑥𝑎* + *𝑥𝑏* ≤ 1 for each edge {*𝑎, 𝑏*} of the conflict graphs *𝐺𝑣*. Note that one could replace Constraints [(3)](#_bookmark14) by individual constraints

However, the maximum clique based formulation (which uses fewer constraints) is stronger than the edge based formulation. It implies all edge based constraints, and the solution space of the linear program- ming relaxation is smaller. This observation was confirmed in prelim- inary numerical experiments: Much larger memory requirements and higher solution times were observed for the edge based formulation when compared to the maximum clique based formulation.

* 1. *Greedy heuristic*

We propose a greedy heuristic which iterates over all mobility de- mands and selects a feasible (i.e., non-conflicting) mobility offer with minimum cost in each iteration. The set of already selected offers

during iteration is denoted by *𝑂*′ *⊂ 𝑂*. For a demand *𝑑* ∈ *𝐷* without

any selected offer, its offers eligible for selection are given by *𝐿𝑑* =

∣ ∀*𝑣* ∈ *𝑂*′ ∶ (*𝑜, 𝑣*) ∉ *𝐸*)

*𝑜*

*𝑑*

cated by the asterisk ∗, the offers *𝐴*4, *𝐶*3, and *𝐵*2 do not require any

{ ∈ *𝑂*

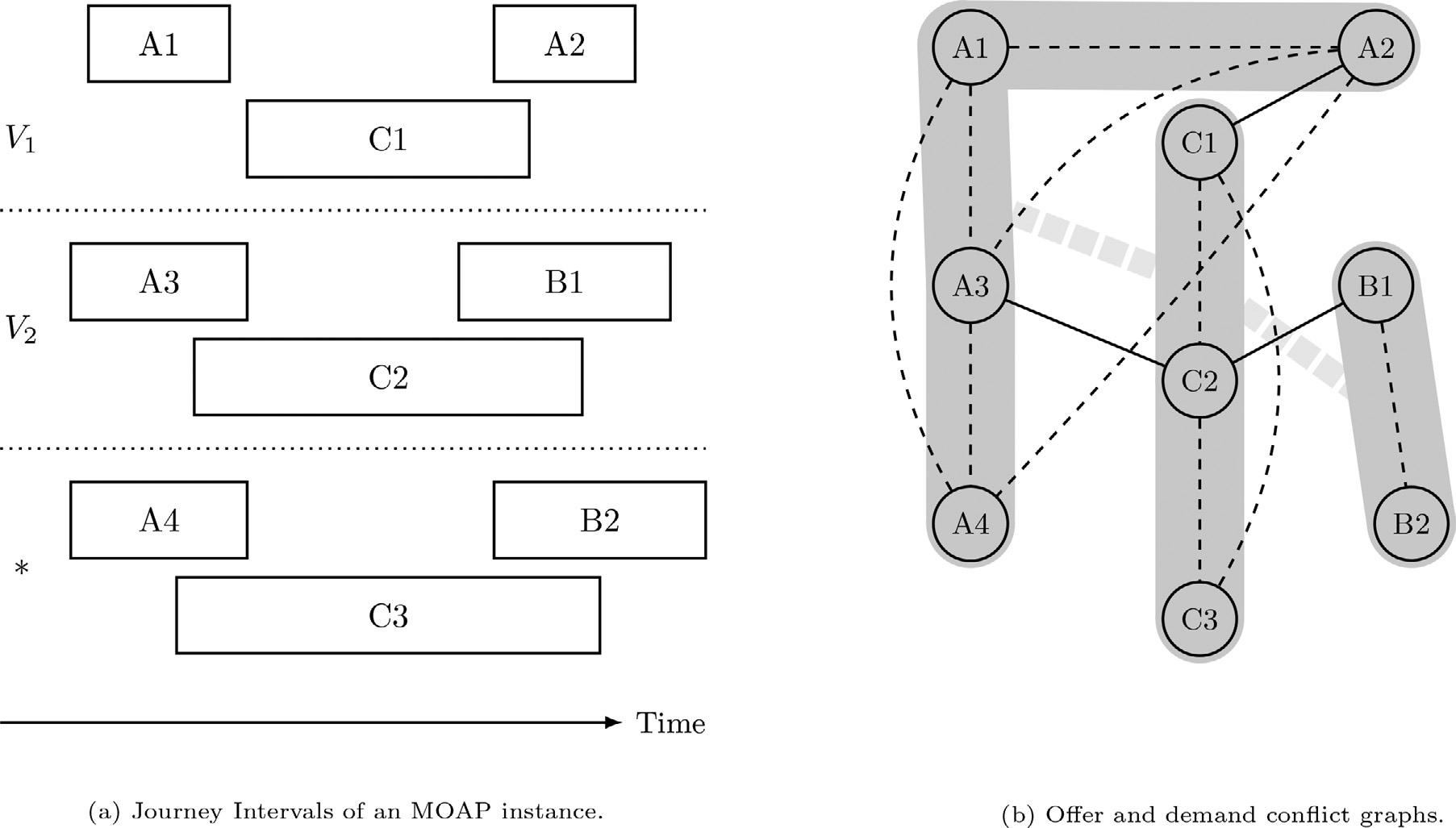
}

. The offer to select is arbitrarily chosen

vehicles. Thus, there are no vehicle induced conflicts in between them. For the offers which require a vehicle, we observe overlapping journey

from argmin*𝑜*∈*𝐿𝑑* (*𝑐𝑜* ). Hence, the order of the iteration is an important

choice which strongly influences the overall objective value. Therefore,



**Fig. 1.** An example showing vehicles, demands, offers, journey intervals, and corresponding conflict graphs.

we propose the following sort criteria where a mobility demand *𝑙* ∈ *𝐷*

is chosen before a mobility demand *𝑟* ∈ *𝐷* if

MinMinCost: min *𝑐𝑜 <* min *𝑐𝑜* ,

* 1. *Adaptive large neighborhood search*

We propose an *adaptive large neighborhood search* (ALNS) metaheuris- tic, whose foundation has originally been introduced by [Shaw (1998)](#_bookmark90).

*𝑜* ∈ *𝑂𝑙*

*𝑜* ∈ *𝑂𝑟*

It was further developed by [Pisinger and Ropke (2010)](#_bookmark77) who introduced

MaxMinCost: min *𝑐𝑜 >* min *𝑐𝑜* ,

an adaptive choice of its operators in [Ropke and Pisinger (2006a)](#_bookmark82).

*𝑜* ∈ *𝑂𝑙 𝑜* ∈ *𝑂𝑟*

MinMinCostPerTime: min *𝑐𝑜*

*𝑜* ∈ *𝑂𝑙 𝜏𝑜*

MaxMinCostPerTime: min *𝑐𝑜*

*𝑜* ∈ *𝑂𝑙 𝜏𝑜*

*<* min *𝑐𝑜* ,

*𝑜* ∈ *𝑂𝑟 𝜏𝑜*

*>* min *𝑐𝑜* ,

*𝑜* ∈ *𝑂𝑟 𝜏𝑜*

Large neighborhood search approaches have already been success- fully used for heuristically solving combinatorial optimization prob- [lems in the domain of vehicle routing (Prescott-Gagnon et al., 2009;](#_bookmark78)

[Ribeiro](#_bookmark84) [and Laporte, 2012), pickup-and-delivery problems (Ropke and](#_bookmark78)

MinAveCost: 1 ⋅ ∑ *𝑐*

*<*  1 ⋅ ∑ *𝑐* ,

[Pisinger, 2006b), or scheduling problems (](#_bookmark84)[Godard](#_bookmark80) [et al., 2005).](#_bookmark84)

|*𝑂𝑙* |

*𝑜*

*𝑜* ∈ *𝑂𝑙*

|*𝑂𝑟* |

*𝑜*

*𝑜* ∈ *𝑂𝑟*

An outline of the adaptive large neighborhood search is shown in

MaxAveCost: 1 ⋅ ∑ *𝑐*

*>*  1 ⋅ ∑ *𝑐* .

[Algorithm 1](#_bookmark18) . The ALNS utilizes *destroy* and *repair* operators, Ω− and

|*𝑂𝑙* |

*𝑜*

*𝑜* ∈ *𝑂𝑙*

|*𝑂𝑟* |

*𝑜*

*𝑜* ∈ *𝑂𝑟*

Random: The ordering is determined by random sampling without

replacement.

This leads to seven different variants of the greedy algorithm. They are used stand-alone, for initial solution generation, and as a part of the repair methods of the ALNS. In detail, the greedy offer selection algorithm works as follows.

**Algorithm 2.** This algorithm greedily selects one mobility offer *𝑜* ∈ *𝑂*

for each of the given mobility demands *𝐷* using the offer conflict graph

*𝐺* = (*𝑂, 𝐸*).

1. Sort all mobility offers *𝑜* ∈ *𝑂* lexicographically: First, by the position

introduced above; Second, ascending by the cost *𝑐𝑜* of the offer. of the corresponding demand according to one of the sorting criteria

1. Mark each offer as *selectable*.
2. Scan the list of sorted offers:

If the current offer *𝑜* ∈ *𝑂* is selectable:

* + Report the offer *𝑜* as selected.
  + Mark all offers *𝑜*′ with { *𝑜*′} ∈ *𝐸* as not selectable.

*𝑜,*

[Algorithm 2](#_bookmark19) has a runtime complexity of U(|*𝑂*| ⋅ log|*𝑂*| + |*𝐸*|) since the offers *𝑂* are sorted and iterated once, and each edge *𝑜, 𝑜*′ ∈ *𝐸* is

{ }

touched at most once.

[**Algorithm 1:** Adaptive large neighborhood search (Pisinger and Ropke, 2010).](#_bookmark77)

**Input**: feasible solution *𝑠*

**Output**: best found solution *𝑠*b

**1** *𝑠*b = *𝑠*; *𝜌*− = (1*,* … *,* 1); *𝜌*+ = (1*,* … *,* 1);

**2 repeat**

**3** select destroy and repair operators *𝑑* ∈ Ω− and *𝑟* ∈ Ω+ using

*𝑝*− and *𝑝*+;

**4** *𝑠*t = *𝑟*(*𝑑*(*𝑠*));

**5 if** *accept(𝑠*t *, 𝑠*) **then**

**6** *𝑠* = *𝑠*t ;

**7 end**

**8 if** *𝑐*(*𝑠*t ) *< 𝑐*(*𝑠*b) **then**

**9** *𝑠*b = *𝑠*t ;

**10 end**

**11** update *𝜌*− and *𝜌*+;

**12 until** *stop criterion is met*;

**13 return** *𝑠*b;

Ω+, respectively. It starts by creating an initial solution *𝑠* followed by a series of destroy and repair moves to improve solutions. A new solution

*𝑠*t is obtained from a previous solution by a move composed of a pair of

destroy and repair operations. This aims at improving a given solution

by unassigning a set of decisions variables (destroy) and subsequently reassigning them (repair). Whenever such a move is *accepted*, e.g., if it improves the current solution, it is the new incumbent solution for the

next iteration. In our ALNS implementation a solution *𝑠* = (*𝑜*1 *,* … *, 𝑜*|*𝐷*|)

is represented as a list of selected offers with *𝑜𝑖* ∈ *𝑂𝑖* , ∀*𝑖* = 1*,* … *,* |*𝐷*|. The

objective value *𝑐*(*𝑠*) = ∑ *𝑐* of a solution *𝑠* is the total cost of its offers.

Demand Conflict Graph Destroy The idea of this operator is to re- move offers from those demands which potentially affect each other. For this purpose, the demand conflict graph described in [Section 3.3](#_bookmark10) is used. For choosing the nodes to be deselected, a start node is chosen uniformly at random. From this start node, a breadth-first search is started until the number of visited nodes

*𝑜*∈*𝑠 𝑜*

des

des

While the standard LNS has only one destroy and one repair operator,

each iteration, one of each operators is chosen based on *weights 𝜌*− and the ALNS extension allows for multiple destroy and repair operators. In

*𝜌*+ assigned to the destroy and repair operators, respectively. The de-

stroy operator *𝜔* ∈ Ω− is chosen with probability *𝑝*− = *𝜌*−∕(∑ − *𝜌*−).

is equal to the number of offers to remove *𝑟*a = ⌈*𝑟* |*𝐷*|⌉ or all

nodes of the connected component of the start node are visited. If all nodes in the connected component are visited but the number of offers to remove has not yet been reached, another start node

from another connected component is chosen and the breadth-

*𝑖 𝑖 𝑖*

*𝑖*∈Ω *𝑖*

The repair operator is chosen analogously. While the LNS accepts only improving solution candidates, for the acceptance criterion of the ALNS [we use a *simulated annealing* based approach as suggested in Ropke and](#_bookmark82)

[Pisinger (2006a). A generated solution *𝑠*t is accepted with probability](#_bookmark82)

*𝑒𝑥𝑝*(− *𝑐*(*𝑠*t )−*𝑐*(*𝑠*) ), where *𝑇 >* 0 is the temperature. The temperature starts with an initial value *𝑇*start and decreases after each iteration to *𝑇* ∶= *𝑐* ⋅ *𝑇* [by using a cooling rate *𝑐* with 0 *< 𝑐 <* 1. As described in Ropke and](#_bookmark82)

*𝑇*

[Pisinger (2006a), setting the start temperature is crucial for the algo-](#_bookmark82) rithm’s performance, but depends on the problem instance. Therefore, the same method as in [Ropke and Pisinger (2006a)](#_bookmark82) is used. The start temperature is set such that the first generated solution after the ini-

tial solution that is *𝑤* percentage worse is accepted with probability *𝑝*w,

where *𝑤* and *𝑝*w are parameters of the ALNS. The weights of the selected destroy and repair operators with indices *𝑖* and *𝑗* are adjusted after each

iteration by setting *𝜌*− = *𝜆𝜌*− + (1 − *𝜆*)*𝜎* and *𝜌*+ = *𝜆𝜌*+ + (1 − *𝜆*)*𝜎*. The pa-

cedure is repeated until *𝑟*des nodes are visited. Then, the selected first search is started anew beginning from this node. This pro-

offer of the corresponding demand of each visited node is dese- lected.

a

*4.3.2. Repair operators*

Two different variants of repair operators are used. One is based on the ILP model introduced in [Section 4.1](#_bookmark12) and the other on the greedy heuristic described in [Section 4.2](#_bookmark15).

Exact Repair The exact repair operator uses the ILP model from [Section 4.1](#_bookmark12) and chooses the offers to select exactly. Assume

*𝑂*′ *⊂ 𝑂* are the still selected offers. Then, we add the constraints

*𝑥𝑜* = 1 ∀ *𝑜* ∈ *𝑂*′ to the model and re-solve it using the mixed inte-

ger linear programming solver.

rameter

*𝑖 𝑖*

*𝑗 𝑗*

Greedy Repair The greedy repair method is based on the greedy

*𝜆* with 0 *< 𝜆 <* 1 is a *decay* parameter determining the impact of the previous weight value. The value *𝜎* modifies the weight depending

the following way (*𝜎*1 *, 𝜎*2 , and *𝜎*3 are parameters of the ALNS): on the performance of the destroy and repair operation pair. It is set in

heuristic introduced in [Section 4.2](#_bookmark15). By using [Algorithm 2](#_bookmark19) for the repair method as in the initial solution generation, the algorithm would always end up in the same solution no matter which offers

* *𝜎* = *𝜎*1

if *𝑐*(*𝑠*t ) *< 𝑐*(*𝑠*b), i.e., the new solution candidate improves the

b

the parameter 0 *< 𝑟*rep *<* 1 which determines the size of a *restricted* are removed. Therefore, a randomization is introduced guided by

best found solution *𝑠* .

* *𝜎* = *𝜎*2 if *𝑐*(*𝑠*b) *< 𝑐*(*𝑠*t ) *< 𝑐*(*𝑠*), i.e., the new solution candidate im-

proves the current solution.

* *𝜎* = *𝜎*3 if *𝑠*t is accepted but *𝑐*(*𝑠*) *< 𝑐*(*𝑠*t ), i.e., the new solution does

improve neither the best nor the current solution but is still accepted

due to the acceptance criterion.

Additionally, *𝜎* = 0 if the new solution candidate has already been

generated, i.e., is a duplicate. Duplicate checking is implemented by us-

ing a hash set storing all hashes of generated solution candidates. Fur- thermore, since the exact repair method (see [Section 4.3.2](#_bookmark20)) is expected

to consume much more time than the greedy repair method, *𝜎* is scaled

by the time needed for performing the corresponding repair operation.

This reduces the bias towards strong but time-consuming operators. Cor- responding destroy and repair operators are described in the following sections.

* + 1. *Destroy operators*

The destroy operators are parametrized by a relative size 0 *< 𝑟*des *<* 1, All destroy operators deselect a certain number of selected offers.

which determines the amount of mobility offers to be deselected. We propose three different approaches:

Random Destroy This operator deselects mobility offers which are chosen uniformly at random from the set of selected offers.

Time Interval Destroy This operator deselects offers within a cer- tain time interval. The idea is to remove offers whose journey in- tervals are close to each other or overlapping. The absolute size

of the time interval is *𝑟*des = ⌈(max*𝑜*∈*𝑂 𝑏𝑜* − min*𝑜*∈*𝑂 𝑎𝑜* ) ⋅ *𝑟*des⌉ and

a

the start of the interval *𝑡𝑎* is chosen uniformly at random from

[min*𝑜*∈*𝑂 𝑎𝑜 ,* max*𝑜*∈*𝑂 𝑏𝑜* ]. In case *𝑡𝑎* + *𝑟*des *>* max*𝑜*∈*𝑂 𝑏𝑜* , the time in-

a

terval goes beyond the considered time horizon. Then, this part

zon, and also comprises the time interval [min*𝑜*∈*𝑂 𝑎𝑜 ,* min*𝑜*∈*𝑂 𝑎𝑜* + of the time interval starts from the beginning of the time hori-

*𝑡𝑎* + *𝑟*des − max*𝑜*∈*𝑂 𝑏𝑜* ]. All mobility offers whose journey intervals

a

overlap the chosen time interval are deselected.

*candidate list* (RCL), commonly used in the context of a *Greedy Randomized Adaptive Search Procedure* ([Feo and Resende, 1995](#_bookmark79)).

Assume we are given an ordering (*𝑑*1*,* … *, 𝑑𝑘* ) with *𝑘* ≤ |*𝐷*| of the

demands which have not been assigned an offer yet. Instead of

choosing the demands in exactly this order, the demand to con- sider next is chosen from the RCL, which is composed of the next

⌈*𝑟*rep|*𝐷*|⌉ demands, uniformly at random. Then, like in the greedy

heuristic, the cheapest feasible offer for this demand is chosen.

Each sort criterion for the demands from [Section 4.2](#_bookmark15) can be used.

# Vehicle classes

Often, one would expect a corporate fleet of vehicles to include many vehicles that could be used interchangeably. This assumption allows to eliminate symmetries not exploited in the approaches presented so far in this paper. [Section 5.1](#_bookmark22) provides an extended problem description which introduces vehicle classes. [Section 5.2](#_bookmark23) proposes an adapted ILP model which makes use of this additional information. Since this adapted ILP model proved to be very eﬃcient in computational experiments (see [Section 6](#_bookmark24)), no additional heuristic methods which exploit the informa- tion on vehicle classes were developed. In [Section 5.3](#_bookmark28), this modeling is compared to the MOAP of [Section 2](#_bookmark3) and it is discussed in which cases the underlying assumption of vehicle interchangeably applies in practice.

* 1. *Modified problem description*

additional input, we are given a set of *vehicle classes 𝑊* . A given mapping The problem description of [Section 2](#_bookmark3) is modified as follows. As an

*𝜑* ∶ *𝑉* → *𝑊* assigns each vehicle *𝑣* ∈ *𝑉* to a vehicle class *𝜑*(*𝑣*) ∈ *𝑊* . All

vehicles of the same class must be indistinguishable, i.e., there must exist

an journey interval and cost preserving bijection between the offers of two arbitrary vehicles from the same class. All mobility offers requiring a

vehicle from the same class are subsumed in an *abstract mobility oﬀer 𝑜* ∈

*𝑂̄*, assigned to the vehicle class *𝑤𝑜* ∈ *𝑊* ∪ {∗}, replacing the vehicle *𝑣𝑜* ∈

*𝑉* assigned to each offer in the original problem definition. As before,

*𝑤𝑜* =∗ denotes that an abstract mobility offer *𝑜* ∈ *𝑂̄* does not require any

vehicle. The problem is to select exactly one abstract mobility offer for

and to choose for each selected offer *𝑜* ∈ *𝑂̄* with *𝑤𝑜* ≠∗ a vehicle *𝑠𝑜* ∈ each demand such that the total cost of the selected offers is minimal,

*𝜑*−1(*𝑤𝑜* ) while overall feasibility is ensured. Feasibility is given if for

each pair *𝑜, 𝑝* ∈ *𝑂̄* of selected offers assigned to the same vehicle *𝑠𝑜* =

*𝑠𝑝* ∈ *𝑉* , it holds that *𝑇𝑜* ∩ *𝑇𝑝* = ∅, i.e., the journey intervals of all selected

offers that use the same vehicle do not overlap. We refer to this problem

as the *Mobility Oﬀer Allocation Problem with Vehicle Classes* (MOAPVC).

* 1. *Integer linear programming model*

The ILP model introduced in the following selects abstract mobility offers. It is ensured that the number of simultaneous selections of ab- stract mobility offers that use the same vehicle class does not exceed the number of vehicles available in that class. From the problem definition, it follows directly that vehicles of the same class have identical conflict

graphs. Thus, we obtain a conflict graph *𝐺𝑤* for each vehicle class *𝑤*.

the conflict graph *𝐺𝑤* of a vehicle class *𝑤* ∈ *𝑊* by *𝐶 𝑤*. Decision vari- Analogously to [Section 4.1](#_bookmark12), we denote the set of maximum cliques in ables *𝑥𝑜* ∈ {0*,* 1} determine whether an abstract mobility offer *𝑜* ∈ *𝑂̄* is selected. The number of vehicles in a vehicle class *𝑤* ∈ *𝑊* is determined

by |*𝜑*−1(*𝑤*)|.

In practice, the identification of vehicles to be used interchangeably depends not only on distinguishing features such as the number of seats, trunk size, cost, energy consumption, or others, but also on the possibil- ity for users to choose mobility options based on such features. In addi- tion to that, vehicle specific appointments for technical inspections or maintenance prevent interchangeability. An interesting use-case where vehicle interchangeability does not apply are car sharing systems where each vehicle is bound to one fixed location. There, although many ve- hicles might be of exactly the same type, most users would accept only vehicles located conveniently, e.g., close to their home address. This choice differs for individual users (i.e., demands), thus preventing ve- hicle interchangeability. If vehicle classes are not given as an explicit input but are still comprised in many problem instances, one could try to detect them automatically, e.g., following the linear time approach of [Lueker and Booth (1979)](#_bookmark59) for deciding interval graph isomorphisms.

# Experimental evaluation

All algorithms presented in [Section 4](#_bookmark9) are implemented in Java 1.8. The general purpose mixed integer linear programming solver IBM ILOG CPLEX Optimizer, version 12.6.2, is used for solving the ILP model. All numerical experiments were conducted using one core of an Intel Xeon 2643 machine with 3.3 GHz and 16 GB RAM each running Linux CentOS 6.5.

|

|

∑

*𝑚𝑖𝑛*

|

|

*𝑥𝑜* ⋅ *𝑐𝑜* (5)

* 1. *Instances*

*𝑠.𝑡.*

*𝑜* ∈ *𝑂̄*

∑

*𝑜* ∈ *𝑂̄𝑑*

*𝑥𝑜* = 1 ∀ *𝑑* ∈ *𝐷* (6)

In order to evaluate the presented solution approaches, two sets of instances are generated. The first set of instances, denoted as *AG*, is created randomly and the created mobility offers have no connection to

∑

*𝑜* ∈ *𝐾*

*𝑥* ≤ |*𝜑*−1 |

| |

*𝑜* |

(*𝑤*)

|

∀ *𝑤* ∈ *𝑊 ,* ∀ *𝐾* ∈ *𝐶 𝑤*

(7)

real-world data. The second set of instances, denoted as *RW*, is based, to a certain degree, on real-world statistical data and some assumptions about travel behavior.

There are several structural differences in the two instance sets which

*𝑥𝑜* ∈ {0*,* 1} ∀ *𝑜* ∈ *𝑂̄*

(8)

reflects the generality of the proposed model. Instance set AG contains

The objective function [(5)](#_bookmark25) remains unchanged and minimizes the sum of the costs of chosen offers. As before, constraints [(6)](#_bookmark26) ensure that for each mobility demand exactly one offer is chosen. Constraints

[(7)](#_bookmark27) prevent that more vehicles than available are used at the same time. A solution of this ILP model only provides an assignment of selected offers to vehicle classes, but not to individual vehicles. From a solu- tion of this ILP, an assignment to individual vehicles can be computed as follows. For each vehicle class, we consider the interval graph that contains the journey intervals of all selected abstract mobility offers which require that vehicle class. Assigning offers to vehicles then cor- responds to finding minimum colorings of these interval graphs with colors corresponding to vehicles. The polynomial-time algorithm of

[Gupta et al. (1979)](#_bookmark85) provides an eﬃcient procedure for this task.

In practical scenarios and in current literature there often exists a vehicle hierarchy. This results in upgradable vehicles such that a vehi- cle of a higher class can be used instead of one of the requested class. This feature is implicitly covered by this modeling approach by gener- ating additional offers for each vehicle class higher than the requested class. Additionally, the costs of these offers for higher classes should be increased to impede their selection.

*5.3. Discussion*

In application scenarios where vehicle interchangeability is given, the ILP model proposed in this section can help to compute solutions more eﬃciently. In cases where vehicle interchangeability is not given, methods from [Section 4](#_bookmark9) can be applied. Note that instances of the MOAP and the MOAPVC can be directly transformed into each other by either omitting vehicle class information or by introducing artificial vehicle classes consisting of single vehicles. Thus, both problem definitions are equally general.

multiple, non-overlapping time windows for mobility demands so that multiple offers for same vehicles may have different journey intervals. This enables modeling alternative dates for one demand. Instance set RW, on the other hand, does not allow alternative dates but the journey intervals of the mobility offers are considered more realistically based on spatial, demographic, and economic data from Vienna, Austria. The start and end time of the mobility offers consider the overall difference in the demand over the time of the day and also differentiate between weekdays and weekends.

The data for both instance sets and the source code of the instance [generators are made publicly available (https://github.com/ait-energy/ seamless). This sections gives an overview of the rather complex in-](https://github.com/ait-energy/seamless) stance generation procedures. We refer to [Appendix A](#_bookmark47) for a more de- tailed description of the instance set RW. In the following, determining a random number according to a discrete uniform distribution over in-

tegers [*𝑎, 𝑏*] is denoted by ∼ *𝐷𝑈* [*𝑎, 𝑏*].

* + 1. *Artificially generated instances (AG)*

First, random instances were created aiming at covering a wide range of scenarios. This instance generation procedure reflects, to some extent, the generality of the proposed modeling. Most parameters of the gener- ator are fixed in order to limit the number of instances. Four parameters are varied which results in an overall number of 144 parameter combi- nations. For each combination, one instance is randomly generated.

The instances aim at representing scenarios with a mixed fleet of ve- hicles and mobility demands representing a variety of situations. A fixed

*number of demands* |*𝐷*| to be generated is chosen from the set {200*,* 1000*,*

2000*,* 5000}. For this number of demands and an expected duration per

*lization rate 𝑃𝑢* is chosen from {20%*,* 40%*,* 60%*,* 80%} and used to deter- offer (derivable from subsequently introduced parameters), a *fleet uti-*

mine the overall number of vehicles in the fleet. Smaller fleet utilization

**Table 1**

Parameters used for generating the 144 random instances of instance set AG.

number of demands |*𝐷*| 200, 1000, 2000, 5000

fleet utilization rate *𝑃𝑢* 20%, 40%, 60%, 80%

vehicle acceptance probability *𝑃𝑎* 0.4, 0.6, 0.8

long demand probability *𝑃𝑙* 0.01, 0.02, 0.05

**Table 2**

AG with the parameters *𝑃𝑎* = 0*.*6 and *𝑃𝑙* = 0*.*02. Instance sizes in numbers of vehicles and offers for instance set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |*𝐷*| = 200 | |*𝐷*| = 1000 | |*𝐷*| = 2000 | |*𝐷*| = 5000 |
| *𝑃𝑢* =|*𝑂*| | 1578 | 21,254 | 76,427 | 445,103 |
| 20% |*𝑉* | | 10 | 36 | 70 | 172 |
| *𝑃𝑢* =|*𝑂*| | 1125 | 11,869 | 41,265 | 229,731 |
| 40% |*𝑉* | | 6 | 18 | 36 | 86 |
| *𝑃𝑢* =|*𝑂*| | 973 | 9036 | 29,958 | 157,753 |
| 60% |*𝑉* | | 4 | 12 | 24 | 58 |
| *𝑃𝑢* =|*𝑂*| | 977 | 7836 | 23,910 | 123,765 |
| 80% |*𝑉* | | 4 | 10 | 18 | 44 |

rates lead to more vehicles in the fleet. Vehicles are classified into four categories (e.g., representing small car, medium car, large car, and van); an individual *vehicle type cost factor* is assigned to each of them (2, 3, 4, and 7, respectively). The number of vehicles per category is deter- mined by fixed *vehicle category portions* (15%, 35%, 35%, 15%) of the overall number of vehicles. Then, for each generated mobility demand, a *minimum vehicle category* is randomly determined according to this dis- tribution. Only vehicles of this or a higher category can be used to fulfill

offers are generated with a *vehicle acceptance probability 𝑃𝑎* chosen from this mobility demand. For each vehicle in a suitable category, mobility

{0*.*4*,* 0*.*6*,* 0*.*8}. Vehicles are chosen as suitable offers with that probabil-

ity. Possible choices between multiple dates for the same appointment

are included by creating multiple offers with different journey intervals

is determined by a *number of journey intervals* ∼ *𝐷𝑈* [1*,* 3]. for the same vehicle. This number of created mobility offers per vehicle

noting hours. Mobility demands are generated for a *planning horizon 𝐻* All dates in the generated instances are represented as integers de- of four weeks (*𝐻* = 24 ⋅ 7 ⋅ 4 = 672). Then, with a *long demand probability*

*𝑃𝑙* chosen from {0*.*01*,* 0*.*02*,* 0*.*05}, it is determined if the demand is consid-

*duration* ∼ *𝐷𝑈* [7*, 𝐻* ] is chosen; otherwise, a base duration ∼ *𝐷𝑈* [1*,* 6] ered to be “long”. For a demand to be considered as “long” means a *base*

is chosen. This base duration of a demand predominantly determines the duration of the journey interval of its offers. For each mobility offer to be generated (i.e., each journey interval and vehicle), a *relative start*

*date* is chosen from ∼ *𝐷𝑈* [2*,* 168]. Finally, the cost of a mobility offer

tor ∼ *𝐷𝑈* [10*,* 30]. This is then used to determine the cost of each offer by is determined by choosing, once for each demand, a cost per time fac-

multiplying it with the duration of its journey interval and the relative cost of the category of the used vehicle.

Supplementary to offers using the considered fleet, mobility offers representing the utilization of public transportation or taxis are in- cluded. Thus, suitable offers which do not require any vehicle are gen- erated. For reflecting the usage of a taxi, or alternatively the regret cost of not fulfilling a demand at all, an additional mobility offer is generated for each mobility demand. Its costs are calculated based on the cost of the mobility offer (of the same demand) with the mini-

*centage* ∼ *𝐷𝑈* [300*,* 600] of the base cost. With a *public transportation* mum vehicle category. The cost of this offer is set to a *taxi cost per-*

*probability* of 0.5, an additional offer is created representing a public

*portation cost percentage* ∼ *𝐷𝑈* [100*,* 300] of the base cost. transportation based route. The cost of this offer is set to a *public trans-*

[Table 1](#_bookmark29) provides an overview of all varying instance generation pa- rameters, which yield an overall number of 144 possible combinations. [Table 2](#_bookmark31) shows the number of offers and vehicles for the 16 instances

generated with a vehicle acceptance probability of *𝑃𝑎* = 0*.*6 and a prob- ability for long demands *𝑃𝑙* = 0*.*02.

* + 1. *Instances based on real-world requirements (RW)*

The second set of instances is based on spatial, demographic, and economic data of Vienna, Austria. The instance generation is based on a defined set of transport modes which are categorized into *foot, public transport, bike, battery electric vehicle (BEV)* and subtypes corresponding to specific car models, *internal combustion engine vehicle (ICEV)* and sub- types corresponding to the size of the vehicle, and *taxi*. Each of these

modes is defined by attributes like CO2 emissions per distance, cost per

time and distance, amount of additional time needed for setup (e.g., get-

ting to the car, time needed for parking) which together defines a cost function.

structed with |*𝑃* | employees and a number of vehicles for each transport For each single benchmark instance an artificial company is con- mode with limited resources depending on *𝑃* and given by the instance parameter *𝜈* ∈ [0*,* 1]. For each transport type with limited availability such as cars and bikes, there are ∼ *𝐷𝑈* [0*,* ⌊*𝜈𝑃* ⌋] such vehicles avail- able. For each person *𝑝* ∈ *𝑃* , a typical work week with work-related and

of *𝑝*. Based on the personal preferences of *𝑝* regarding mode of transport private events is constructed which forms the set of mobility demands

(which depend on statistical data, e.g., regarding gender and probability of owning a driving license), for each acceptable transport mode, one mobility offer is created for each resource of that mode with the corre- sponding journey interval and cost. The journey interval depends on the time given by the mobility demand and the travel and setup time of the corresponding mode of transport. The cost consists of three factors: the

cost given by the distance, the cost given by CO2 emissions, and the cost

given by the time. While the costs for distance and CO2 are fixed for a

specific mobility demand and mode of transport, the costs for time are

only considered for business mobility demands and based on average salaries.

For each combination of company size |*𝑃* | ∈

{500*,* 750*,* 1000*,* 1250*,* 1500*,* 1750} and relative number of vehicles

*𝜈* ∈ {0*.*05*,* 0*.*10*,* 0*.*15} 30 instances are created. [Table 3](#_bookmark32) provides an

overview of the average size of the instances regarding the number of

offers and number of vehicles. [Appendix A](#_bookmark47) provides a more detailed description of this instance generation procedure.

* 1. *Computational results of the ILP model*

Extensive computational experiments using both instance sets were performed. For determining CPLEX parameters, we used the built-in pa- rameter tuning tool on the set of training instances also used for ALNS parameter tuning (see [Section 6.3](#_bookmark34)) with a time limit of one hour. It turned out that the default parameters of CPLEX worked best. [Tables 4](#_bookmark33) and [5](#_bookmark35) show results obtained by the exact solution approach using the ILP model from [Section 4.1](#_bookmark12) for instance set AG and RW, respectively. In [Table 4](#_bookmark33) results are aggregated over the instance generation parameters

|*𝐷*|, *𝑃𝑢*, *𝑃𝑎*, and *𝑃𝑙* and in [Table 5](#_bookmark35) the results are grouped by the instance

parameters |*𝑃* | and *𝜈*. A time limit of at most one hour per instance

Columns “#S” provide the number of solved instances. Columns “gap” was used. The number of instances per group is given in columns “#I”.

provide the average optimality gap between the objective function value of the found solution and the lower bound in percent. Columns “#N/S” provide the number of instances in which no feasible solution was found,

either because of the time or the memory limit. Columns “*𝑡*[s]” provide

optimality. For the instance set RW, columns “*𝑡*∗[s]” show the average the average runtime in seconds over all instances that were solved to

time needed to solve the instances with the ILP model utilizing vehicle classes as described in [Section 5.2](#_bookmark23). Note that the optimal solution value for both variants with and without vehicle classes remains equal and that the time needed for computing actual vehicle assignments based on the vehicle class assignments is negligible.

**Table 3**

Instance sizes in average numbers of vehicles and offers for instance set RW.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | *𝜈* = 0.05 | *𝜈* = 0.1 | *𝜈* = 0.15 |  |  | *𝜈* = 0.05 | *𝜈* = 0.1 | *𝜈* = 0.15 |
| |*𝑃* | = 500 | |*𝑂*| | 83783.6 | 175242.2 | 264657.2 | |*𝑃* | = 1250 | |*𝑂*| | 526631.2 | 980385.6 | 1462976.7 |
|  | |*𝑉* | | 71.1 | 152.0 | 225.3 |  | |*𝑉* | | 190.8 | 350.2 | 566.7 |
| |*𝑃* | = 750 | |*𝑂*| | 177830.6 | 368634.7 | 537975.3 | |*𝑃* | = 1500 | |*𝑂*| | 747897.3 | 1580761.5 | 2245212.8 |
|  | |*𝑉* | | 100.8 | 209.1 | 332.9 |  | |*𝑉* | | 227.1 | 472.6 | 682.9 |
| |*𝑃* | = 1000 | |*𝑂*| | 349469.7 | 643205.2 | 942703.7 | |*𝑃* | = 1750 | |*𝑂*| | 940197.0 | 1936257.1 | 3085115.9 |
|  | |*𝑉* | | 155.7 | 293.5 | 427.8 |  | |*𝑉* | | 249.8 | 538.4 | 820.4 |

**Table 4**

Computational results of the exact solution approach using the ILP model for instance set AG.

Aggregated by |*𝐷*| Aggregated by *𝑃𝑙*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| |*𝐷*| | #I | #S | gap | *𝑡*[s] |  | *𝑃𝑙* | #I | #S | gap | *𝑡*[s] |
|  | 200 | 36 | 36 | 0.00% | *<* 1 |  | 0.01 | 48 | 36 | 6.94% | 1054 |  |
|  | 1000 | 36 | 36 | 0.00% | 100 |  | 0.02 | 48 | 32 | 11.13% | 1295 |  |
|  | 2000 | 36 | 22 | 0.13% | 1652 |  | 0.05 | 48 | 30 | 20.88% | 1465 |  |
|  | 5000 | 36 | 4 | 51.81% | 3333 |  |  |  |  |  |  |  |
| Aggregated by *𝑃𝑢* Aggregated by *𝑃𝑎* | | | | | | | | | | | | |
| *𝑃𝑢* | | #I | #S | gap | *𝑡*[s] | *𝑃𝑎* | | #I | #S | gap | *𝑡*[s] | |
| 20% | | 36 | 31 | 2.78% | 802 | 0.4 | | 48 | 33 | 8.91% | 1240 | |
| 40% | | 36 | 21 | 22.54% | 1648 | 0.6 | | 48 | 31 | 13.05% | 1329 | |
| 60% | | 36 | 22 | 17.62% | 1420 | 0.8 | | 48 | 34 | 17.00% | 1244 | |
| 80% | | 36 | 24 | 9.00% | 1215 |  | |  |  |  |  | |

In instance set AG, all small and medium instances (|*𝐷*| ≤ 2000) are either solved to optimality or the obtained solution shows a very small

gap. Only four large instances (|*𝐷*| = 5000) are solved to optimality. The

quite poor. For *𝑃𝑢* = 20 % and for *𝑃𝑢* = 80 %, more instances are solved to large gaps indicate that the solution found for the larger instances are

utilization rates (*𝑃𝑢* = 20 %) are easier to solve because they have less optimality or only a small gap is obtained. Instances with lower vehicle

(*𝑃𝑢* = 80 %) have fewer feasible solutions which makes them easier to conflicts between offers. Instances with higher vehicle utilization rates solve as well. The impact of the parameters *𝑃𝑙* and *𝑃𝑎* is less clear. Long

instances with large values of *𝑃𝑙* are harder to solve. Instances with lower mobility demands increase the number of conflicts which explains that values for *𝑃𝑎* seem to be easier, which might result from fewer decision

variables.

In instance set RW, CPLEX was able to solve all instances with

|*𝑃* | ∈ {500*,* 750} to optimality within the time limit of one hour. The larger instances with |*𝑃* | ≥ 1250 are harder to solve. The diﬃculty in- creases with |*𝑃* | but also with *𝜈*. The large optimality gaps for these

ficient heuristic algorithm. For the largest instance set with |*𝑃* | = 1750 instances and the high runtimes show the necessity of a fast and ef- and *𝜈* ≥ 0*.*10, CPLEX could not even find any feasible solution in 8 cases.

When using vehicle classes instead, all instances can be solved within a few seconds. As described before, this model has much less variables and constraints and can therefore be solved faster. Interchangeable ve-

hicles only exist for instances RW, therefore we only report the results for this instance class.

* 1. *Computational results of the ALNS*

For evaluating the (A)LNS and its operators proposed in [Section 4.3](#_bookmark17) we first have to select appropriate parameters. We choose [them by using the parameter tuning tool *irace* (López-Ibáñez et al., 2016) which iteratively samples the parameter space, evaluates the](#_bookmark57) samples, and discards them if a Friedman test shows that it is dom- inated by other parameter configurations. Therefore, we generate a

[Section 6.1.2](#_bookmark30) with |*𝑃* | = 500 and *𝜈* ∈ {0*.*05*,* 0*.*10*,* 0*.*15}. For each of these new training set of instances based on instance set RW described in

3 combinations, 10 instances are created resulting in a training set consisting of 30 instances. On these instances, irace is executed with a total of 20,000 runs and a time limit of 1 minute per run. To get more detailed data about the performance of the different LNS configurations, one run of irace is performed for each destroy operator (*Random, Time Interval*, and *Demand Conflict Graph*). Based on these results, one run of irace is performed for the ALNS utilizing all three destroy operators

each with *𝑟*des = 0*.*15 and with the two repair operators, *exact* repair

and *greedy* repair with sorting criterion *MaxMinCost* and *𝑟*rep = 0*.*1. The

parameter space in which irace operates is shown in [Table 6](#_bookmark36).

Note that the *𝜎*-values may seem counter-intuitive at first glance because The resulting parameter configurations are shown in [Tables 7](#_bookmark38) and [8](#_bookmark39).

*𝜎*1 *< 𝜎*2 *< 𝜎*3. However, this can be explained by the following observa- tions: First, a higher *𝜎*2 and *𝜎*3 value encourages diversification espe-

cially at the beginning of the algorithm because then it is more likely to accept a non-improving solution candidate. Second, for the exact repair

operator the value of *𝜎*3 is irrelevant because the generated solution

struction. Therefore, a higher *𝜎*3 value encourages the use of the greedy candidate cannot be worse than the solution candidate before the de-

repair method which is more likely to improve the solution at the begin- ning of the algorithm. In the remaining section, we refer to these values when mentioning the *Random, Time Interval, Demand Conflict Graph*, or *ALNS* configuration.

As expected, the performance of the greedy algorithms strongly de- pends on the sorting criterion that is used. [Fig. 2](#_bookmark37) compares the greedy algorithms on instance set AG normalized by relative differences to the best solution obtained by all algorithms for the corresponding instance. Additionally, we compare our proposed greedy algorithms to the greedy algorithm *G1-MW* proposed in [Ng et al. (2014)](#_bookmark66), where *G1-MW* is the

**Table 5**

Computational results of the exact solution approach using the ILP model for instance set RW.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *𝜈* | #I | |*𝑃* | | #S | #N/S | gap | *𝑡*[s] | *𝑡*∗[s] | |*𝑃* | | #S | #N/S | gap | *𝑡*[s] | *𝑡*∗[s] |
| 0.05 | 30 |  | 30 | – | 0.0% | 23 | 1 |  | 26 | – | 0.6% | 1593 | 3 |
| 0.1 | 30 | 500 | 30 | – | 0.0% | 55 | 1 | 1250 | 15 | – | 38.7% | 2575 | 3 |
| 0.15 | 30 |  | 30 | – | 0.0% | 70 | 1 |  | 15 | – | 51.9% | 2639 | 3 |
| 0.05 | 30 |  | 30 | – | 0.0% | 131 | 1 |  | 18 | – | 43.3% | 2445 | 3 |
| 0.1 | 30 | 750 | 30 | – | 0.0% | 330 | 1 | 1500 | 7 | – | 60.6% | 3350 | 4 |
| 0.15 | 30 |  | 30 | – | 0.0% | 299 | 1 |  | 8 | – | 75.5% | 3150 | 4 |
| 0.05 | 30 |  | 30 | – | 0.0% | 651 | 2 |  | 10 | – | 45.7% | 3060 | 4 |
| 0.1 | 30 | 1000 | 26 | – | 0.4% | 1238 | 2 | 1750 | 5 | 2 | 57.3% | 3352 | 5 |
| 0.15 | 30 |  | 28 | – | 42.1% | 1728 | 2 |  | 2 | 6 | 75.4% | 3395 | 5 |

**Table 6**

Parameter tuning scenario.

Description Possible Values

Repair method {*greedy, exact*}

The relative amount of solution parts that are destroyed *𝑟*des [0.1,0.9] Sort criterion for greedy repair {*MinMinCost, MaxMinCost, MinMinCostPerTime, MaxMinCostPerTime, MinAveCost, MaxAveCost* }

Relative size of the RCL *𝑟*rep for the greedy repair [0.1,0.9]

*𝜎*1*, 𝜎*2*, 𝜎*3 [1,100]

Decay parameter *𝜆* [0.01,0.99]

Initial temperature control parameter *𝜔* [0.00001, 0.1]

Initial temperature acceptance parameter *𝑝𝜔* [0.01,0.99]

Cooling rate *𝑐* [0.1,0.999999]

**Fig. 2.** Comparison of different greedy con- struction heuristics for instance set AG.





**Table 7**

LNS Parameter tuning results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Destroy method (fixed) | Repair method | Sort criterion | *𝑟*rep | *𝑟*des |
| Random | *greedy* | *MaxMinCost* | 0.1308 | 0.1580 |
| Time Interval | *greedy* | *MaxMinCost* | 0.1063 | 0.1053 |
| Demand Conflict Graph | *exact* | – | – | 0.1836 |

**Table 8**

ALNS Parameter tuning results.

Method *𝜎*1 *𝜎*2 *𝜎*3 *𝜆 𝜔 𝑝𝜔 𝑐*

ALNS 23 40 50 0.2377 0.0373 0.656 0.2267

[best performing heuristic evaluated in that paper (see](#_bookmark66) [Table 2](#_bookmark31)[, (Ng et al., 2014)). The greedy algorithm *G1-MW* always chooses the cheapest feasi-](#_bookmark66) ble offer from all remaining, not yet satisfied demands. *MaxMinCost* out- performs all other sorting criteria and yields a starting solution of about

10% worse than the best found solution on average. *MinMinCost, Min-*

*MinCostPerTime, MinAveCost*, and *Random* perform poorly and are omit-

ted for readability. *MaxAveCost* and *MaxMinCostPerTime* cannot com- pete with *MaxMinCost*. Algorithm *G1-MW* also does not perform well. We assume this is mainly because it is too greedy in the sense that it does not consider the consequences of always choosing the currently

cheapest offer. The computational time per instance for all variants of the greedy algorithms is always below one second per instance.

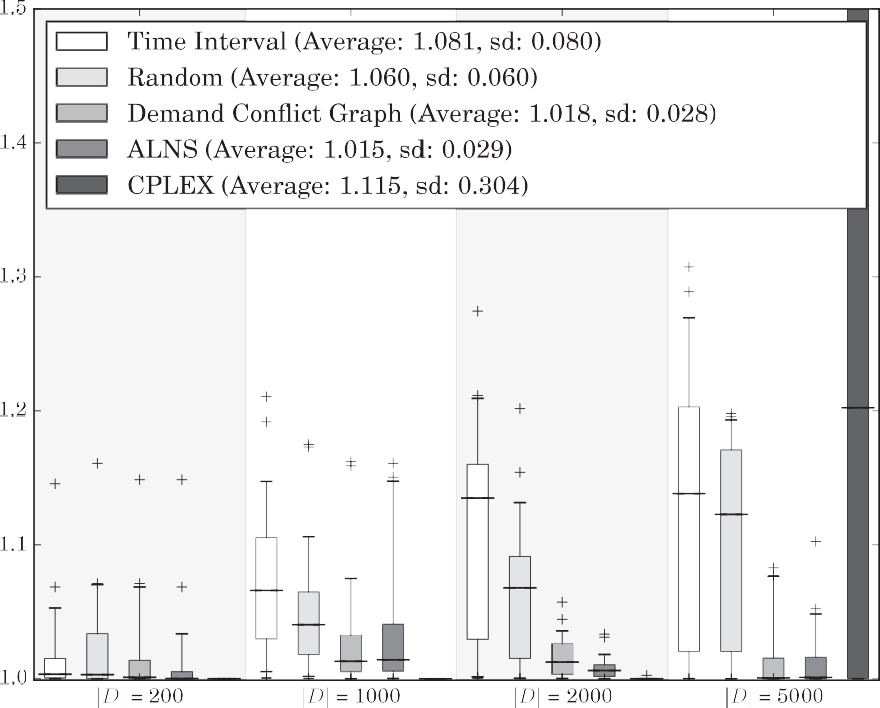
To compare all proposed approaches at a glance, [Figs. 3](#_bookmark40) and [4](#_bookmark41) show a comparison of the (A)LNS configurations *Random, Time Interval, De- mand Conflict Graph*, and *ALNS* along with the results obtained by CPLEX for instance sets AG and RW, respectively. All runs of the (A)LNS in the following tests are executed with a time limit of 5 minutes each. Re- garding the results of instance set AG, for most of the small and medium

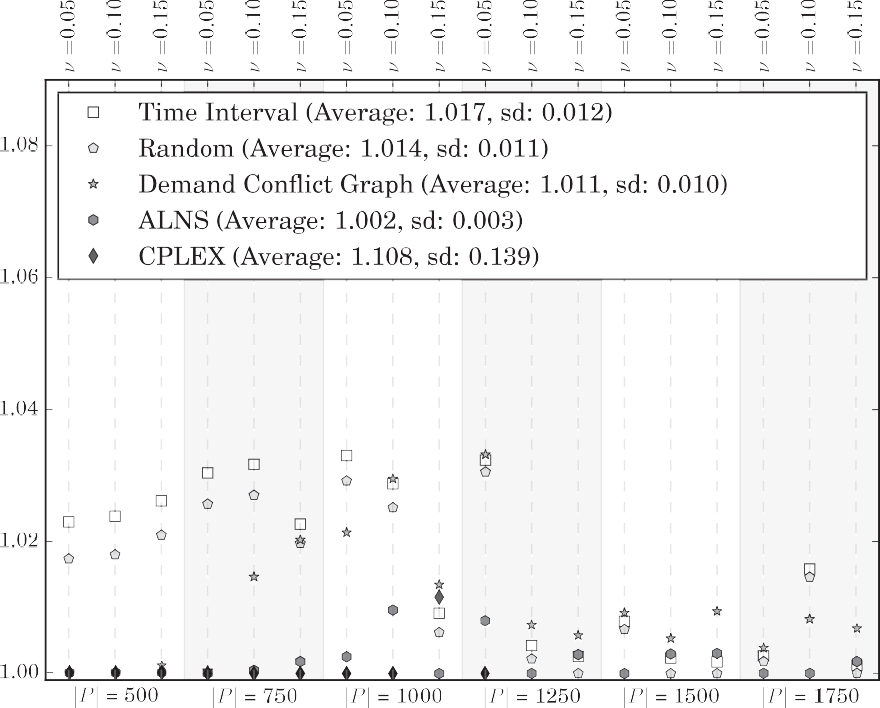
the large instances. While on the smallest instances with |*𝐷*| = 200 most instances CPLEX finds exact solutions, but it does not perform well on

configurations perform more or less equally good, CPLEX outperforms the other approaches for the medium instances. The largest instances can be solved best by the LNS configuration *Demand Conflict Graph* and the *ALNS* which both are on average within 1.5-1.8% of the best found solution. Note that the bad results by CPLEX on the larger instances come from the time limit of 1 h, after which the execution is halted and the best feasible solution is reported. *Random* and *Time Interval* have similar performance although *Random* seems to give better results on average by about 2.1%. Overall, depending on the available computa- tional time and the instance to be solved, either CPLEX or the *ALNS* is the best suited choice.

values are aggregated over the instances with the same value for |*𝑃* | [Fig. 4](#_bookmark41) shows the results on instance set RW in which the numerical and *𝜈* by computing the mean of the relative differences to the best

found objective values. These results confirm the conclusions from the

**Fig. 3.** Comparison of the different destroy op- erators of the LNS, the ALNS, and CPLEX for instance set AG.

**Fig. 4.** Comparison of the ALNS, the different destroy operators of the LNS, and the results obtained by CPLEX for instance set RW, nor- malized by best found solution. Additionally,

per set and sorted by *𝜈* within their category the results are aggregated over the 30 instances regarding |*𝑃* | in ascending order.

results of instance set AG to some extent. In these instances, CPLEX as well is able to solve the small to medium instances and the ALNS seems to be the best choice for larger instances when comparing the average value over all instances. For the larger instances, CPLEX and also the LNS configuration *Demand Conflict Graph* are in some cases not able to

with |*𝑃* | ≤ 1000 the CPLEX model can be used if the run-time is not that find feasible solutions. Therefore, we conclude that for smaller instances

important. On the other hand, for larger instances and when short run- times are required, the ALNS is recommended. Note that although the relative differences are much smaller in this instance set compared to the

**Table 9**

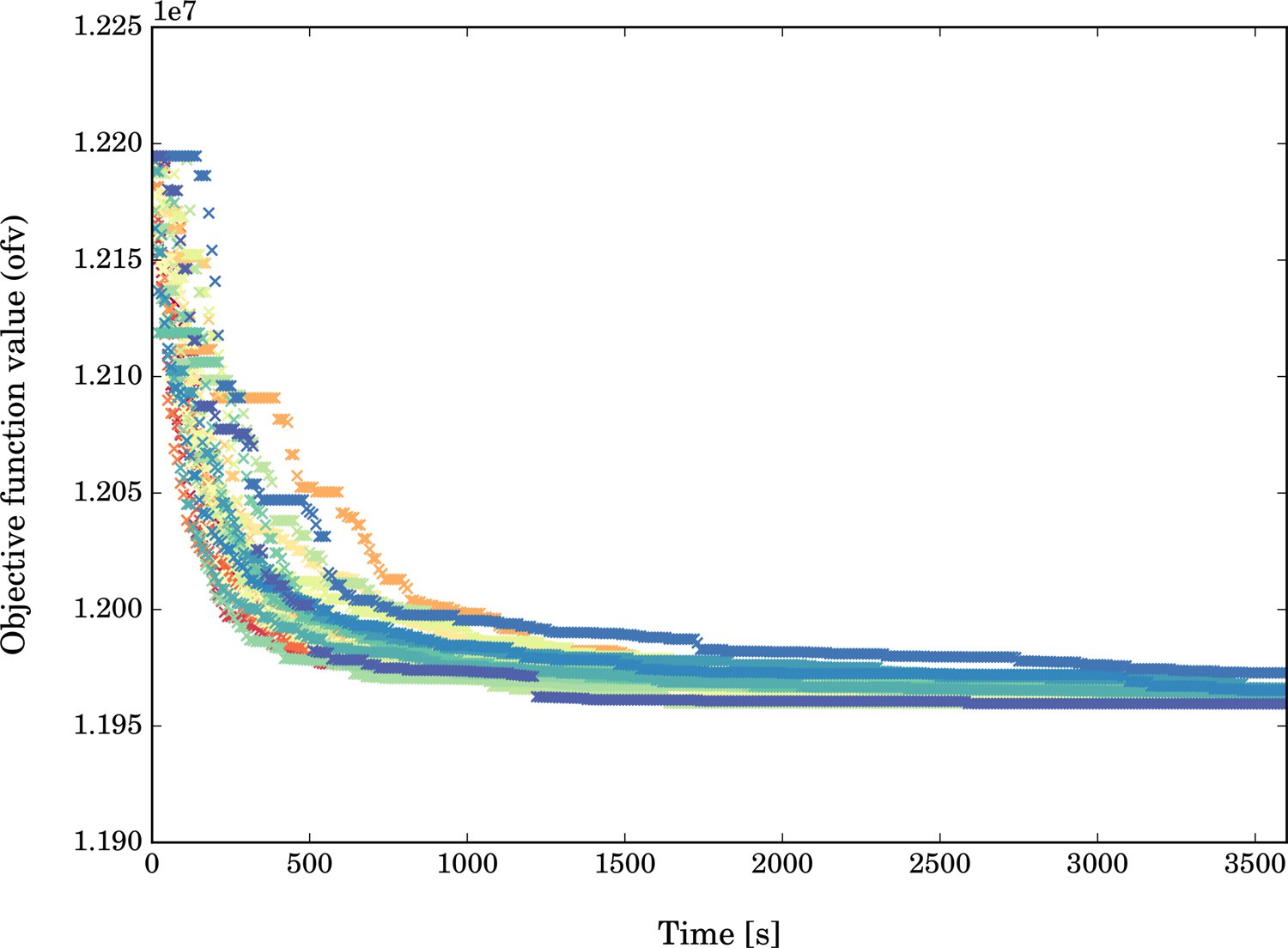
Method recommendations for different instance sizes.

fast runtime required (*<* 1s) fast runtime not required

|*𝑃* | ≤ 1000 or |*𝐷*| ≤ 2000 *greedy-MaxMinCost CPLEX*

|*𝑃* | *>* 1000 or |*𝐷*| *>* 2000 *greedy-MaxMinCost ALNS*

previous, the absolute values of the objective function are much larger due to the structure of the instances. To summarize, [Table 9](#_bookmark42) recommends

**Fig. 5.** Typical progression of the objective value over time for the ALNS in 30 indepen- dent runs of

the instance *E1250\_10* with *𝜈* = 0*.*10.

methods for different instance sizes based on the number of employees or the number of demands.

To assess the convergence behavior of the ALNS, [Fig. 5](#_bookmark43) shows the decrease over time of the objective function value for 30 independent

**Table 10**

Fleet size, vehicle utilization, and data about the number of trips using shared vehicles for instance set AG.

Aggregated by |*𝐷*| Aggregated by *𝑃𝑙*

*𝑢*

*𝑢*

runs of the ALNS on one instance. Therefore, the currently best objective

values of each run are collected in discrete time intervals of 10 seconds

|*𝐷*| |*𝑉* | *𝑃* ∗

*𝑉 𝑢 𝑃𝑙*

|*𝑉* | *𝑃* ∗

*𝑉 𝑢*

200 6.7 36.53% 66.69% 0.01 25.9 47.60% 65.81%

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| over the whole run-time of one hour. We observe a much higher vari- | 1000 | 32.0 | 49.06% | 72.06% | 0.02 | 38.0 | 47.79% | 72.19% |
| ance between runs during the initial phase of the search. Even the worst | 2000 | 44.7 | 50.19% | 75.58% | 0.05 | 73.9 | 45.08% | 81.19% |
| runs after about 1000 seconds outperform the best runs before 200 sec- | 5000 | 109.3 | 51.53% | 77.92% |  |  |  |  |
| onds. Thus, performing multiple independent runs with the goal of ob- Aggregated by *𝑃𝑢* Aggregated by *𝑃𝑎*  *𝑃* |*𝑉* | *𝑃* ∗ *𝑉 𝑢 𝑃* |*𝑉* | *𝑃* ∗ *𝑉 𝑢* | | | | | | | | |
| taining a better overall result can be recommended only if the available | *𝑢*  20% | 86.7 | *𝑢*  92.06% | | *𝑎*  0.4 | 45.9 | *𝑢*  45.48% 70.35% | |
| time per run is suﬃciently high. Additionally, we evaluated the time | 40% | 44.2 | 47.06% 79.19% | | 0.6 | 45.9 | 46.98% 73.48% | |
| needed to find the best solution within a runtime of 1000 seconds for | 60% | 30.0 | 52.58% 66.44% | | 0.8 | 45.9 | 48.02% 75.35% | |

30.64%

the instances RW. The results showed that for the smaller (E500, E750) and larger (E1500, E1750) instances the best solution was found in only about 44.81% of the runtime and for the medium (E1000, E1250) in- stances in about 68.08% of the runtime. This indicates that for these instances the ALNS is usually able to converge to its final value in be- tween 7 and 11 min of runtime.

Finally, we analyze the properties of the results regarding their ap-

lization of the shared fleet vehicles *𝑃* ∗ and the relative number of trips plication to practical use cases. Therefore, we computed the average uti- using one of these vehicles *𝑉 𝑢*. For the average utilization, we computed

*𝑢*

the total reservation times of the vehicles and compared these to the whole time horizon of the specific instance. We show these values for instance set AG in [Table 10](#_bookmark44) and for instance set RW in [Table 11](#_bookmark46). In the latter table we additionally compare the changes of the traditional fleet

size to the shared fleet size Δ*𝑉* and the change in the vehicle utilization

rates Δ*𝑃* ∗. To compute these values we assumed a classical one-to-one assignment of vehicles to employees, i.e., in the instances with |*𝑃* | we

*𝑢*

assumed a fleet size of 500.

rate than in instance set RW and increasing with higher values of |*𝐷*| The results of instance set AG show a generally higher utilization and |*𝑃𝑢*|. Since the instance generation procedure of the instance set AG

does not distinguish between day and night trips, the demand is more evenly distributed which results in the increase of the utilization rate. Also, the utilization increases with a higher demand to vehicle ratio and clearly with an increasing input vehicle utilization ratio. Considering the

80% 22.8 57.03% 54.56%

results of instance set RW, we see a reduction of the fleet size between 50 and 85% while the average utilization rates are increasing in most cases. There are, however, some cases in which the vehicle utilization rates are actually lower than in the case of fixed vehicle assignments. This is because the goal of the algorithm is not to maximize the vehicle usage but to find the most cost-effective mobility offer allocations. This means that in many cases the alternative offers, e.g., public transport, are actually better for the company and are therefore chosen over a shared fleet vehicle. The same effect can also be observed when looking at the number of offers using fleet vehicles which are also rather low. If desirable, the objective function of the optimization problem could

utilization values *𝑃* ∗ and *𝑉 𝑢*. In this work, however, we did not pursue be replaced by a utility maximization function in order to increase the

*𝑢*

this variant because we believe in the benefit of offering alternative mobility offers to employees.

# Conclusion and outlook

This paper introduces the *Mobility Oﬀer Allocation Problem* for cor- porate mobility services and solution algorithms to solve it. We pro- pose a methodology that integrates a mixed fleet of vehicles with other mobility options such as public transportation or taxis. An experimen-

**Table 11**

Fleet size, vehicle utilization, and data about the number of trips using shared vehicles for instance set RW.

*𝑢 𝑢 𝑢 𝑢*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *𝜈* | |*𝑃* | | Δ*𝑉* | *𝑃* ∗ | Δ*𝑃* ∗ | *𝑉 𝑢* | |*𝑃* | | Δ*𝑉* | *𝑃* ∗ | Δ*𝑃* ∗ | *𝑉 𝑢* |
| 0.05 |  | -85.78% | 59.33% | 8.51% | 22.93% |  | -84.74% | 58.50% | 8.29% | 23.27% |
| 0.1 | 500 | -69.60% | 51.80% | 0.97% | 31.67% | 1250 | -71.98% | 53.87% | 3.66% | 30.17% |
| 0.15 |  | -54.94% | 43.87% | -6.96% | 21.90% |  | -54.66% | 52.66% | 2.45% | 31.10% |
| 0.05 |  | -86.56% | 60.17% | 9.45% | 22.23% |  | -84.86% | 58.70% | 8.42% | 23.63% |
| 0.1 | 750 | -72.12% | 53.47% | 2.75% | 35.23% | 1500 | -68.49% | 54.89% | 4.62% | 30.07% |
| 0.15 |  | -55.61% | 45.20% | -5.52% | 32.50% |  | -54.47% | 52.17% | 1.89% | 26.33% |
| 0.05 |  | -84.43% | 57.80% | 7.47% | 24.57% |  | -85.73% | 58.50% | 8.22% | 21.83% |
| 0.1 | 1000 | -70.65% | 52.13% | 1.81% | 31.53% | 1750 | -69.23% | 52.80% | 2.62% | 29.53% |
| 0.15 |  | -57.22% | 49.83% | -0.49% | 33.23% |  | -53.12% | 53.63% | 3.44% | 27.38% |

tal evaluation shows the trade-offs of the proposed solution methods regarding computational times and solution quality for different kinds of instances. The results demonstrate the applicability of the methods for realistic instance sizes and show performance indicators interesting for real world applications. For improving the proposed adaptive large neighborhood search, designing better destroy operators seems worth investigating. Currently, when choosing demands to be destroyed, the selected offers are not taken into account. Including the potential cost savings for a demand might lead to more eﬃcient operators. In practice, such approaches must be applied in a dynamic setting where demands arrive over time and offers are booked in advance. There, the idea of delaying assignment decisions in order to increase planning flexibility provides further practical benefits, e.g., in case of vehicle breakdowns

# Appendix A. Detailed description of the generation of instance set RW

The benchmark instances are based on demographic, spatial, and economic data of Vienna, Austria, and consider a company which oper-

ates in that area. First, a set of mode of transport classes *𝐾* are defined

consisting of the following types: *Foot, Public transport, Bike, Battery elec-*

*tric vehicle (BEV)* and subtypes corresponding to specific car models, *In- ternal combustion engine vehicle (ICEV)* and subtypes corresponding to the size of the vehicle and *Taxi*. For both BEV and ICEV several sub- categories are defined which correspond to car models, e.g., for BEVs

ties of each *𝑘* ∈ *𝐾* are the following: we consider Smart ED, Nissan Leaf, and Mitsubishi iMiev. The proper-

or late returns. We believe the proposed approaches are applicable in

such rolling horizon scenarios. However, further evaluating and adapt- ing them in such settings is a relevant direction of future research.

Parameter

Domain Unit Description

The conflict graph based modeling of the problem facilitates the in- clusion of additional constraints. For example, a consideration of per- sons, potentially involved in multiple appointments, can be included

*𝑐*

*𝜖𝑘* ℝ g/km CO2 emissions per distance unit

*𝑣𝑘* ℝ m/s average speed

*𝑘*

via additional conflict edges. Another possible extension would be to *𝑑*

suggest multiple employees to share a vehicle if their requests are simi- *𝑘*

*𝑐*

*𝑡*

ℝ 1/km cost in Euro per distance

ℝ 1/min cost in Euro per time

lar, e.g., [Enzi et al. (2020)](#_bookmark69). This could be modeled by introducing offers

that satisfy more than one demand. A major limitation of the proposed modeling is that mobility offers refer to fixed journey intervals. In case the sequencing of offers becomes relevant, one would obtain a variant of the well-known Vehicle Routing Problem. Though journey intervals cannot, locations actually can vary in the presented modeling. Assum- ing a future scenario with a fleet of self-driving vehicles, each mobility demand could state a fixed start and end location specifying a request for a ride between given locations taking a fixed amount of time. Then, two offers using the same vehicle are in conflict if driving from the end location of the earlier offer to the start location of the later offer is not possible within the given time. So, an adapted conflict graph based mod- eling could prove useful also for such scenarios. The approach also aims at fostering the use of battery electric vehicles by helping to achieve uti- lization rates required for compensating high purchase prices. Recharg- ing processes can either be included simplistically, by prolonging the journey intervals of mobility offers, or, a more detailed modeling could combine ideas from this work and that of [Sassi and Oulamara (2017)](#_bookmark87), where battery loading states are modeled explicitly.

Overall, we believe the proposed modeling provides a flexibility that offers a range of interesting applications not restricted to corporate envi- ronments, e.g., a large housing unit equipped with a fleet of cars shared by the inhabitants. A larger scale application would be to implement the approach for station based car-sharing providers.

# Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence

*𝑎𝑘* ℝ s additional time needed for setup (e.g., getting to

the car, time needed for parking)

Then, a company is constructed consisting of one or more depots

Δ *⊂ 𝐿*, where each *𝛿* ∈ Δ is represented by its geographic coordinate and

*𝐿* is the set of all possible locations. The company has a set of employees

*𝑃* , and a number of available instances *𝑛𝑘* of each transport class *𝑘* ∈ *𝐾*. Note that *𝑛𝑘* = ∞ for *foot, public transport*, and *taxi*.

Each employee *𝑝* ∈ *𝑃* has a gender *𝜃𝑝* ∈ {f *,* m}, a hierarchy status

*ℎ𝑝* ∈ {b*,* m*,* w} (boss, middle management, worker), an associated oﬃce location *𝛿𝑝* ∈ Δ, a home location *𝑙𝑝* ∈ *𝐿*, a work start time *𝜏𝑠* ∈ ℕ, and a work end time *𝜏𝑒* ∈ ℕ For all *𝑘* ∈ *𝐾* it is specified if employee *𝑝* is willing to accept offers using transport mode *𝑘*, denoted by *𝜔𝑝𝑘* ∈ {0*,* 1},

∀*𝑘* ∈ *𝐾, 𝑝* ∈ *𝑃* .

Then, for each employee *𝑝* ∈ *𝑃* on each day *𝑡* ∈ *𝑇* of the considered time horizon *𝑇* an ordered list of events *𝐸𝑝𝑡* = (*𝑒𝑝𝑡,* … *, 𝑒𝑝𝑡*) is generated

0 *𝑛*

(representing a working day of this employee) consisting of the follow- ing attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Parameter | Domain | Unit | Description |  |
| *𝛼𝑒* | ℕ | min | latest arrival (in number of minutes from the start |  |
|  |  |  | of the time horizon) |  |
| *𝛽𝑒* | ℕ | min | earliest departure |  |
| *𝑠𝑒* | ℕ | min | service duration |  |
| *𝑙𝑒* | *𝐿* |  | location |  |
|  | *𝑡𝑒* | {w*,* m*,* p*,* h} |  | activity type: *work, meeting, private, home* |  |

Furthermore, for each pair of locations *𝑙*1*, 𝑙*2 ∈ *𝐿* a distance *𝑑𝑘* , travel

*𝑖𝑗*

time *𝑡𝑘* , and cost matrix *𝑐𝑘* is computed for each *𝑘* ∈ *𝐾* based on the

*𝑖𝑗 𝑖𝑗*

the work reported in this paper.

route from *𝑙*1 to *𝑙*2 in the road network.

*Value Settings* This section describes how the independent values of

the variables described above are set. Some of the variables are chosen

Parameter Variability

Value

randomly following the stated probability distribution. In these cases the actual instance is generated by drawing one sample of each of these distributions.

*Transport classes:*

Scope Value

Parameter Variability

*𝜖𝑘* fixed all average values of the respective car category

*𝑣𝑘* fixed all foot: 5, bike: 16, car: 30, public transport: 20

[km/h]

*𝑘* fixed all total cost of ownership divided by total km

*𝑐*

*𝑑*

*𝛼𝑒* fixed private activity: at any time outside working hours. Work

meeting: at any time within the working hours.

*𝛽𝑒* fixed *𝛼𝑒* + *𝑠𝑒*

*𝑠𝑒* fixed private meetings in the morning 60 minutes, in the

minutes based on probability distribution P *𝑠𝑒* . evening 120 minutes. Work meetings between 30 and 180

*𝑙𝑒* fixed based on P *ℎ* for private activities, on P *𝑜* for work meetings

*𝑡𝑒* fixed for each day: private activity in the morning with 20%

number of work meetings is based on *ℎ𝑝* which results in a probability, in the evening with 65% probability. The

average amount of time spent in meetings. A meeting is inserted into the daily schedule of the employee until this time is spent or it does not fit in anymore.

*Distance, travel time, and cost:*

*𝑘* fixed all average gross salary in Austria including

*𝑐*

*𝑡*

additional costs for employer

*𝑎𝑘* fixed all foot: 0, bike: 120, car: 600, public transport: 300,

Parameter Variability

Value

*Company:*

taxi: 300 [s]

*𝑘*

*𝑖𝑗*

*𝑑*

fixed Aerial distance between *𝑖* and *𝑗* multiplied by a constant

sloping factor of the respective mode of transport *𝑘*

*𝑑𝑘*

*𝑘*

*𝑡*

*𝑖𝑗*

fixed

*𝑖𝑗*

*𝑣𝑘*

*𝑐𝑘*

fixed *𝑐𝑘𝑑𝑘* + *𝑐𝑘𝑡𝑘* + *𝜖𝑘𝑐𝑒* , where *𝑐𝑒* are the CO2 costs which are set

*𝑖𝑗*

*𝑑 𝑖𝑗*

*𝑡 𝑖𝑗*

to 5 Euro per ton.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Parameter | Variability | Value |  |
| *𝐿* | fixed | geometric centers of all 250 registration districts |  |
|  |  | of Vienna |  |
| *𝑇* | fixed | one week |  |
| Δ | fixed | two locations chosen randomly following the  probability distribution P *𝑜* of *𝐿*, where P *𝑜* is based |  |
|  |  | on statistical data of oﬃce locations in Vienna |  |
| |*𝑃* | | variable | integer value |  |
| *𝜈* | variable | real value in the interval [0,1] determining *𝑛𝑘* ,  ∀ |  |
| *𝑛𝑘* | fixed | for bikes, BEVs, and ICEVs: between 0 and ⌊*𝜈*|*𝑃* |⌋ |  |
|  | *Employee:* |  |  |  |

*Generation of the Mobility Oﬀers*

Based on the data described above we extract mobility demands and offers which form the actual instance of our optimization prob-

lem. First, we generate the set of mobility demands *𝐷* by considering

the events *𝐸𝑝* = ⋃

*𝑡*∈*𝑇*

*𝐸𝑝𝑡* of each employee *𝑝* ∈ *𝑃* . Since we assume that

the company fleet is located at the depots Δ, each mobility demand

*𝑑* ∈ *𝐷* consists of a tour starting and ending at the oﬃce location *𝛿𝑝*

of the corresponding employee *𝑝*. Therefore we construct the set of de-

mands *𝐷𝑝* = {*𝑑𝑝,* … *, 𝑑𝑝* } with *𝑑𝑝* = (*𝑒𝑝, 𝑒𝑝 ,* … *, 𝑒𝑝*) *⊆ 𝐸𝑝* with *𝑞 > 𝑗* for

0 *𝑚*

*𝑒𝑝*

*𝑖*

*𝑒𝑝*

*𝑗 𝑗*+1 *𝑞*

*𝑘* ∈ *𝐾*

all *𝑗* = 0*,* … *, 𝑛* with *𝑡 𝑗* = *𝑡 𝑞* = w, ∀*𝑖* ∈ 0*,* … *, 𝑚* for each employee *𝑝* ∈ *𝑃* . For each *𝑝* ∈ *𝑃* and each *𝑑* ∈ *𝐷𝑝* a set of mobility offers *𝑂𝑑* is created. There is one offer for each transport class *𝑘* ∈ *𝐾* which is accepted by the employee, i.e., for which *𝜔𝑝𝑘* = 1, denoted by *𝑘𝑜* ∈ *𝐾*. Each offer *𝑜* ∈ *𝑂𝑑* has an *journey interval* [*𝑎𝑜, 𝑏𝑜*] with *𝑎𝑜, 𝑏𝑜* ∈ ℝ defining its start time *𝑎𝑜* and

*𝑒𝑝*

end time *𝑏𝑜*. The start time *𝑎𝑜* is given by the latest arrival *𝛼 𝑗*+1 of the first event of the associated demand subtracted by half the setup time 1 *𝑎𝑘*

2

of the corresponding vehicle type *𝑘 𝐾* and the travel time *𝑡𝑘*

∈

*𝑙*1 *,𝑙*2

with

Value

*𝑒𝑝 𝑝*

*𝑒𝑝*

*𝑒𝑝 𝑘*

1 *𝑘*

Parameter Variability

*𝑙*1 = *𝑙 𝑗* = *𝛿*

*𝑒𝑝*

and *𝑙*2 = *𝑙 𝑗*+1 . The end time is given by *𝑏𝑜* = *𝛽 𝑞* + *𝑡𝑙 ,𝑙*

*𝑝*

3 4

+ 2 *𝑎*

*𝜃𝑝* fixed based on demographic data of female and male employees

(f: 46.78%, m: 53.22%)

with *𝑙*3 = *𝑙 𝑞* and *𝑙*4 = *𝛿* resulting in a duration *𝜋𝑜𝑑𝑝* = *𝑏𝑜𝑑𝑝* − *𝑎𝑜𝑑𝑝* .

Finally, the cost *𝑐𝑜* of each offer *𝑜* ∈ *𝑂𝑑* , ∀*𝑑* ∈ *𝐷𝑝, 𝑝* ∈ *𝑃* is generated

based on the cost matrix of the relevant events and the corresponding

transport class *𝑘*. The salary costs which depend on the duration of the

|  |  |  |
| --- | --- | --- |
| *ℎ𝑝* | fixed | *𝑃* (*ℎ𝑝* = b) = 0*.*01, *𝑃* (*ℎ𝑝* = m) = 0*.*1, *𝑃* (*ℎ𝑝* = w) = 0*.*89 |
| *𝛿𝑝* | fixed | chosen uniformly at random out of Δ |

*𝑙𝑝* fixed chosen randomly following the probability distribution P *ℎ*

of *𝐿*, where P *ℎ* is based on statistical data of residential

offer are, however, only considered for *work events*, i.e., the journeys from work to the meetings and from the meetings back to work. More

specifically, the cost of an offer *𝑜* ∈ *𝑂𝑑* with *𝑑𝑝* = (*𝑒𝑝, 𝑒𝑝 ,* … *, 𝑒𝑝*), ∀*𝑝* ∈

*𝑗 𝑗*+1 *𝑞*

locations in Vienna

*𝜏𝑠* fixed chosen randomly following a probability distribution P *𝜏𝑠*

*𝑃* contains the setup costs *𝐶𝑆* and the travel costs *𝐶𝑇* is *𝐶𝑜* = *𝐶𝑆* + *𝐶𝑇*

with:

between 5 and 11 a.m.

*𝜏𝑒* fixed *𝜏𝑠* + amount of daily working hours, which are chosen

randomly following a probability distribution P *𝜏𝑒* which depends on *𝜃𝑝* and *ℎ𝑝*

*𝐶𝑆* = *𝑎𝑘𝑐𝑘*

*𝑞*∑−1

*𝑡*

*𝐶𝑇* =

*𝑐𝑘 𝑝 𝑝*

– 0*.*8*𝑡 𝑝 𝑝 𝑐𝑘*(1 − Γ *𝑝 𝑝*

)*,* with

*𝜔𝑝𝑘* fixed we defined 7 combinations of accepted mode of transports,

e.g., car only, public transport only, mixed. For each

combination at most different acceptance scenarios are

*𝑖*=*𝑗*

*𝑙𝑒𝑖 𝑙𝑒𝑖*+1

{

*𝑙𝑒𝑖 𝑙𝑒𝑖*+1 *𝑡*

*𝑒𝑝 𝑒𝑝*

*𝑒𝑖 𝑒𝑖*+1

*𝑒𝑝*

*𝑒𝑝*

*Events:*

probability distribution P *𝜔* considering gender and the defined. The combinations are chosen randomly based on a probability that *𝑝* has a driving license which itself is based

on statistical data. The acceptance scenario of the chosen category is taken uniformly at random.

Γ *𝑝 𝑝* =

*𝑖 𝑖*+1

*𝑒 𝑒*

1 if (*𝑡 𝑖* = w ∧ *𝑡 𝑖* = m) ∨ (*𝑡 𝑖* = m ∧ *𝑡 𝑖* = w)

0 else

# References

[Angelelli, E., Bianchessi, N., Filippi, C., 2014. Optimal interval scheduling with a resource constraint. Comput. Oper. Res. 51, 268–281.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0001)

[Arkin, E.M., Silverberg, E.B., 1987. Scheduling jobs with fixed start and end times. Discret Appl. Math. 18 (1), 1–8.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0002)

[Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B., 2001. A unified ap- proach to approximating resource allocation and scheduling. J. ACM (JACM) 48 (5), 1069–1090.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0003)

[Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wag-](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0004) [ner, D., Werneck, R.F., 2016. Route planning in transportation networks. In: Algo- rithm Engineering. Springer International Publishing, pp. 19–80.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0004)

[Bates, J., Leibling, D., 2012. Spaced Out: Perspectives on parking policy. Report. RAC Foundation.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0005)

[Betz, J., Werner, D., Lienkamp, M., 2016. Fleet disposition modeling to maximize utiliza- tion of battery electric vehicles in companies with on-site energy generation. Transp. Res. Procedia 19, 241–257.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0006)

[Bunte, S., Kliewer, N., 2009. An overview on vehicle scheduling models. Public Transp. 1 (4), 299–317.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0007)

[Burke, E.K., Mareček, J., Parkes, A.J., Rudová, H., 2010. A supernodal formulation of vertex colouring with applications in course timetabling. Ann. Oper. Res. 179 (1), 105–130.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0008)

[Butman, A., Hermelin, D., Lewenstein, M., Rawitz, D., 2010. Optimization problems in multiple-interval graphs. ACM Trans. Algorithms (TALG) 6 (2), 40.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0009)

[Desfontaines, L., Desaulniers, G., 2018. Multiple depot vehicle scheduling with controlled trip shifting. Transp. Res. Part B 113, 34–53.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0010)

Enzi, M., Parragh, S. N., Pisinger, D., 2020. Modeling and solving a vehicle-sharing prob- lem. [arXiv:2003.08207](http://arxiv.org/abs/2003.08207).

[Ernst, A.T., Gavriliouk, E.O., Marquez, L., 2011. An eﬃcient Lagrangean heuristic for rental vehicle scheduling. Comput. Oper. Res. 38 (1), 216–226.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0012)

[Ernst, A.T., Horn, M., Krishnamoorthy, M., Kilby, P., Degenhardt, P., Moran, M., 2007. Static and dynamic order scheduling for recreational rental vehicles at tourism hold- ings limited. Interfaces 37 (4), 334–341.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0013)

[Escoﬃer, B., Monnot, J., Paschos, V.T., 2005. Weighted coloring: further complexity and approximability results. In: Italian Conference on Theoretical Computer Science. Springer, pp. 205–214.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0014)

[Farber, M., 1984. Domination, independent domination, and duality in strongly chordal graphs. Discret. Appl. Math. 7 (2), 115–130.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0015)

[Feo, T.A., Resende, M.G., 1995. Greedy randomized adaptive search procedures. J. Glob.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0016)

[Optim. 6 (2), 109–133.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0016)

[Godard, D., Laborie, P., Nuijten, W., 2005. Randomized large neighborhood search for cumulative scheduling.. In: International Conference on Automated Planning and Scheduling (ICAPS), 5, pp. 81–89.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0017)

[Goodall, W., Dovey, T., Bornstein, J., Bonthron, B., 2017. The rise of mobility as a service.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0018)

[Deloitte Rev. 20, 112–129.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0018)

[Götz, K., Sunderer, G., Birzle-Harder, B., Deffner, J., Berlin, B., 2011. Attraktivität und akzeptanz von elektroautos: ergebnisse aus dem projekt OPTUM optimierung der umweltentlastungspotenziale von elektrofahrzeugen. ISOE-Studientexte 18.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0019)

[Gupta, U.I., Lee, D.-T., Leung, J.-T., 1979. An optimal solution for the channel-assignment problem. IEEE Trans. Comput. 807–810.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0020)

[Gupta, U.I., Lee, D.-T., Leung, J.-T., 1982. Eﬃcient algorithms for interval graphs and circular-arc graphs. Networks 12 (4), 459–467.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0021)

[Gustin, W., 1963. Orientable embedding of Cayley graphs. Bull. Am. Math. Soc. 69 (2), 272–275.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0022)

[Hassold, S., Ceder, A.A., 2014. Public transport vehicle scheduling featuring multiple ve- hicle types. Transp. Res. Part B 67, 129–143.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0023)

[Hoff,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0024) [A., Andersson, H., Christiansen, M., Hasle, G., Løkketangen, A., 2010. Industrial aspects and literature survey: fleet composition and routing. Comput. Oper. Res. 37 (12), 2041–2061.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0024)

[Koetter,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0025) [F., Ostermann, J., Jecan, D.V., 2015. Schedule rating method based on a fragmen- tation criterion. In: The Fourth International Conference on Advances in Vehicular Systems, Technologies and Applications, p. 28.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0025)

[Kolen,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0026) [A.W., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C., 2007. Interval scheduling: a survey. Nav. Res. Logist. 54 (5), 530–543.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0026)

[Kovalyov,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0027) [M.Y., Ng, C., Cheng, T.E., 2007. Fixed interval scheduling: models, applications, computational complexity and algorithms. Eur. J. Oper. Res. 178 (2), 331–342.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0027)

[Kroon,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0028) [L.G., Salomon, M., Van Wassenhove, L.N., 1997. Exact and approximation algo- rithms for the tactical fixed interval scheduling problem. Oper. Res. 45 (4), 624–638.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0028) [López-Ibáñez,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0029) [M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Per-](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0029)

[spect. 3, 43–58.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0029)

[Lueker,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0030) [G.S., Booth, K.S., 1979. A linear time algorithm for deciding interval graph iso- morphism. J. ACM (JACM) 26 (2), 183–195.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0030)

[Ma,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0031) [H., Balthasar, F., Tait, N., Riera-Palou, X., Harrison, A., 2012. A new comparison be- tween the life cycle greenhouse gas emissions of battery electric vehicles and internal combustion vehicles. Energy Policy 44, 160–173.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0031)

[Marx,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0032) [D., 2004. Graph colouring problems and their applications in scheduling. Periodica Polytech. Electr. Eng. 48 (1–2), 11–16.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0032)

[Metcalfe,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0033) [R., Dolan, P., 2012. Behavioural economics and its implications for transport. J. Transp. Geogr. 24, 503–511.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0033)

[Moon, J.W.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0034), [Moser, L., 1965. On cliques in graphs. Israel J. Math. 3 (1), 23–28](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0034).

[Ng,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0035) [C.T., Cheng, T.C.E., Bandalouski, A.M., Kovalyov, M.Y., Lam, S.S., 2014. A graph-the- oretic approach to interval scheduling on dedicated unrelated parallel machines. J. Oper. Res. Soc. 65 (10), 1571–1579.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0035)

[Oliveira,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0036) [B.B., Carravilla, M.A., Oliveira, J.F., 2017. Fleet and revenue management in car rental companies: a literature review and an integrated conceptual framework. Omega 71, 11–26.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0036)

[Oliveira,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0037) [B.B., Carravilla, M.A., Oliveira, J.F., Toledo, F.M., 2014. A relax-and-fix-based algorithm for the vehicle-reservation assignment problem in a car rental company. Eur. J. Oper. Res. 237 (2), 729–737.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0037)

[Ostermann,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0038) [J., Renner, T., Koetter, F., Hudert, S., 2014. Leveraging electric cross-company car fleets through cloud service chains: the shared e-fleet architecture. In: Global Con- ference (SRII), 2014 Annual SRII. IEEE, pp. 290–297.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0038)

[Pisinger,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0039) [D., Ropke, S., 2010. Large neighborhood search. In: Handbook of Metaheuristics.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0039)

[Springer, pp. 399–419.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0039)

[Prescott-Gagnon, E., Desaulniers, G., Rousseau, L.-M., 2009. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time win- dows. Networks 54 (4), 190–204.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0040)

[Ribeiro,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0041) [G.M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. 39 (3), 728–735.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0041)

[Ropke,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0042) [S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0042)

[Ropke,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0043) [S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0043)

[Sanders,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0044) [P., Schulz, C., 2012. High quality graph partitioning.. Graph Partitioning Graph Clust. 588 (1), 1–17.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0044)

[Sassi,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0045) [O., Oulamara, A., 2017. Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. Int. J. Prod. Res. 55 (2), 519–535.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0045)

[Shaw,](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0046) [P., 1998. Using constraint programming and local search methods to solve vehi- cle routing problems. In: International Conference on Principles and Practice of Con- straint Programming. Springer, pp. 417–431.](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0046)

[Shoup, D., 2017. The High Cost of Free Parking: Updated Edition. Routledge](http://refhub.elsevier.com/S2192-4406(21)00137-4/sbref0047).