Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 21/07

D. Gaul, K. Klamroth and M. Stiglmayr

Event-based MILP models for ride-hailing applications

March 1, 2021

http://www.imacm.uni-wuppertal.de

Event-based MILP models for ride-hailing applications

Daniela Gaul[a,1], Kathrin Klamroth[a], Michael Stiglmayr[a]

[a] *School of Mathematics and Natural Sciences, University of Wuppertal, 42119 Wuppertal, Germany*

Abstract

Ride-hailing services require efficient optimization algorithms to simultaneously plan routes and pool users in shared rides. We consider a static dial-a-ride problem (DARP) where a series of origin-destination requests have to be assigned to routes of a fleet of vehicles. Thereby, all requests have associated time windows for pick-up and delivery, and may be denied if they can not be serviced in reasonable time or at reasonable cost. Rather than using a spatial representation of the transportation network we suggest an event-based formulation of the problem. While the corresponding MILP formulations require more variables than standard models, they have the advantage that capacity, pairing and precedence constraints are handled implicitly. The approach is tested and validated using a standard IP-solver on benchmark data from the literature. Moreover, the impact of, and the trade-off between, different optimization goals is evaluated on a case study in the city of Wuppertal (Germany).

Keywords: ride-hailing, dial-a-ride problem, mixed-integer programming, event-based graph representation, on-demand transportation services, weighted-sum objective

1. Introduction

Cities are increasingly suffering from particulate pollution and congestion. As a consequence, modern concepts for traffic management and transportation planning have to compromise between customer satisfaction and economical and ecological criteria. One way to address these challenges is the concept of ride-hailing: an 'on-demand ride-hailing service' is a taxi-like service, typically operated by mini-buses, where users submit pick-up and drop-off locations via smartphone. Users with similar origin or destination are assigned to the same ride whenever this is economically or ecologically useful. Since rides may be

Email addresses: gaul@math.uni-wuppertal.de (Daniela Gaul), klamroth@math.uni-wuppertal.de (Kathrin Klamroth), stiglmayr@math.uni-wuppertal.de (Michael Stiglmayr)

[1] Corresponding author.

shared, users may have to accept longer ride times. In exchange, they can be offered lower fares.

To encourage people to use ride-hailing services, an efficient and fair route planning is of central importance. From a modelling perspective, the *dial-a-ride problem* (DARP) consists of scheduling transportation requests with given pick-up and delivery times and assigning them to a set of vehicle routes. Common optimization objectives are to accept as many user requests as possible, to minimize the overall routing costs, and to minimize the maximum and/or the total waiting times and ride times. For an individual user request, the following options can be considered:

- the request may be integrated (optimally) in an already existing tour (which can be modified by the new request),
- the request may initiate a new tour (operated by an additional vehicle), or
- the request may be denied.

The DARP is motivated by a variety of real-life applications. Dial-a-ride systems originated as door-to-door transportation, primarily in rural areas, where public transport is rarely available with buses going only a couple of times a day. Another application occurs in health care transport: Elderly, injured or disabled persons are often not able to drive on their own or to use public transport, and dial-a-ride services are a convenient alternative to expensive taxi services. Recently, several so-called 'on-demand ride-hailing services' have emerged. Prominent examples are Uber[2], DiDi Chuxing[3], Lyft[4] or moia[5]. Ride-hailing services complement public transport and are usually established in urban areas. They are an alternative to motorized private transport and thus help to reduce the number of vehicles in the cities. This paper is motivated by a particular ride-hailing service that was established in the city of Wuppertal (Germany) in collaboration with the project bergisch.smart[6]. This ride-hailing service is run by the local public transport provider and is designed to complement the bus service.

In this paper, we focus on the *static* DARP assuming that all user requests are known in advance, i.e. before the DARP is solved. This is in contrast to the *dynamic* DARP where requests are revealed during the day and vehicle routes are updated whenever a new request arrives. While the static DARP is important on its own right and most research focuses on the static DARP, see e. g. Ho et al. (2018), we note that static DARP models can also be extended to the dynamic scenario by using a rolling-horizon strategy.

[2] `https://www.uber.com/de/en/ride/uberpool/`
[3] `https://www.didiglobal.com/travel-service/taxi`
[4] `https://www.lyft.com/rider`
[5] `https://www.moia.io/en/how-it-works`
[6] `https://www.bergischsmartmobility.de/en/the-project/`

2. Literature Review

The following literature review focuses specifically on variants of the static multi-vehicle dial-a-ride problem with time windows. For a broader review the reader is referred to Cordeau and Laporte (2007) who cover research on the DARP up to 2007, and to Molenbruch et al. (2017) or Ho et al. (2018) for more recent surveys.

Early work is carried out by Jaw et al. (1986), who develop one of the first heuristics for the multi-vehicle DARP. Depending on the users' earliest possible pick-up times, the heuristic determines the cheapest insertion position in an existing route in terms of user satisfaction and operator costs. Desrosiers et al. (1991), Dumas et al. (1989) and Ioachim et al. (1995) first identify groups of users to be served within the same area and time. In a second step, these 'clusters' are combined to obtain feasible vehicle routes. This decomposition approach, however, leads in general to suboptimal solutions. Cordeau and Laporte (2003) are the first to apply tabu search to the DARP. In doing so they make use of a simple neighborhood generation technique by moving requests from one route to another. Their reported computational results on real-life data as well as on randomly generated instances validate the efficiency of the heuristic. Recent work on the static multi-vehicle DARP related to tabu search is based on Cordeau and Laporte (2003), see, for example, Detti et al. (2017); Guerriero et al. (2013); Kirchler and Calvo (2013); Paquette et al. (2013). Cordeau (2006) is, to the best of our knowledge, the first to apply a branch-and-cut (B&C) algorithm to the DARP. In addition to valid inequalities derived from the traveling salesman problem, the vehicle routing problem, and the pick-up and delivery problem, new valid inequalities are added as cuts to a mixed-integer linear programming (MILP) formulation of the DARP. While the MILP model of Cordeau (2006) is based on three-index variables, and is a standard formulation of the basic DARP that has been used as well for further problem extensions (see Ho et al. (2018)), Ropke et al. (2007) propose an alternative MILP formulation of the DARP using two-index variables. Furthermore, Ropke et al. (2007) derive three more classes of valid inequalities and combine them with some of the cuts presented by Cordeau (2006) to develop a branch-cut-and-price (BCP) algorithm. Some of these inequalities are employed in more recent works on more complex versions of the DARP, see, for example, Braekers et al. (2014); Parragh (2011). The former work proposes a B&C algorithm to exactly solve small-sized instances as well as a deteministic annealing meta-heuristic for larger problem instances, while Parragh (2011) proposes deterministic annealing and insertion heuristics, respectively. There exist several exact solution approaches that are based on branch-and-bound (B&B) frameworks. For example, Cortés et al. (2010) combine B&C and Bender's decomposition to solve a DARP that allows users to swap vehicles at specific locations during a trip. Liu et al. (2015) devise a B&C algorithm and formulate two different models for a DARP with multiple trips, heterogeneous vehicles and configurable vehicle capacity. They introduce eight families of valid inequalities to strengthen the models. Qu and Bard (2015) propose an exact method based on BCP with heterogeneous vehicles and con-

figurable vehicle types. Gschwind and Irnich (2015) develop a new exact BCP algorithm for DARP which outperforms the B&C algorithm suggested by Ropke et al. (2007). They show that their algorithm is capable of solving an instance with 8 vehicles and 96 users which is, to the best of our knowledge, to date the largest instance of the standard DARP solved by an exact method.

Since exact branching based procedures are usually limited to small to medium-sized instances, there has been some work on hybridizing B&B-like methods with heuristics. For example, Hu and Chang (2014) apply a branch-and-price (B&P) algorithm to a DARP with time-dependent travel times, in which the pricing subproblem is solved by large neighborhood search. As a side result, they observe that the length of time windows can have a significant impact on the size of the fleet, the objective function value, the CPU time, the average ride time or the average pick-up time delay. While exact solution approaches are often based on B&B, at the level of heuristics and meta-heuristics there is a wide set of methods that can be applied. Jorgensen et al. (2007) propose an algorithm which assigns passengers to vehicles using a genetic algorithm. In a second step, routes are constructed sequentially by means of a nearest neighbor procedure. Genetic algorithms are also used by Cubillos et al. (2009), who obtain slightly better results than Jorgensen et al. (2007). Atahran et al. (2014) devise a genetic algorithm within a multi-objective framework. Belhaiza (2017) and Belhaiza (2019) combine genetic crossover operators with variable neighborhood search and adaptive large neighborhood search, respectively. A hybrid genetic algorithm is also developed by Masmoudi et al. (2017), who according to Ho et al. (2018), achieve the second-best results on benchmark instances of the standard DARP. Hosni et al. (2014) present a MILP formulation which is then decomposed into subproblems using Lagrangian relaxation. Feasible solutions are found by modifying the subproblems in a way that generates a minimal incremental cost in terms of the objective function. Gschwind and Drexl (2019) develop a meta-heuristic solution method for the DARP, which is at the moment probably the most efficient heuristic method to solve the standard DARP. They adopt an adaptive large neighborhood method that was developed by Ropke and Pisinger (2006) for the pick-up and delivery problem with time windows. The resulting algorithm integrates a dynamic programming algorithm into large neighborhood search. Souza et al. (2020) present a simple heuristic based on variable neighborhood search combined with a set covering strategy. A heuristic repair method based on a greedy strategy, that reduces the infeasibility of infeasible intital routes is developed by Chen et al. (2020). Their repair method improves about 50% of the fixing ability of a local search operator.

While most of the literature on the DARP focuses on the single-objective DARP, the DARP is examined from a multi-objective optimization perspective as well. As stated by Ho et al. (2018), this line of research can be divided into three main categories: use of a weighted sum objective, a lexicographic approach and determination of (an approximation of) the Pareto front, the set of all non-dominated solutions. Weighted sum objectives have been used e. g. by Jorgensen et al. (2007); Melachrinoudis et al. (2007); Mauri et al. (2009); Kirch-

ler and Calvo (2013) or Bongiovanni et al. (2019). Next to the customers' total transportation time, which is a common objective in single-objective DARPs, the weighted sum objective introduced by Jorgensen et al. (2007) is composed of the total excess ride time, the customers' waiting time, the drivers' work time as well as several penalty functions for the violation of constraints. The weights on these criteria are chosen with respect to the relative importance of the criteria from a user-perspective. While the methods to solve DARP with a weighted sum objective range from a genetic algorithm (Jorgensen et al. (2007)), to tabu search (Melachrinoudis et al. (2007); Kirchler and Calvo (2013)), simulated annealing (Mauri et al. (2009)) or B&C (Bongiovanni et al. (2019)), most of the authors either adapt the weights introduced by Jorgensen et al. (2007) or choose them with respect to individual assessments of their relative importance. Garaix et al. (2010); Schilde et al. (2014) and Luo et al. (2019) consider a lexicographic ordering of the objective functions w.r.t. their relative importance. In this context, Luo et al. (2019) have proposed a two-phase BCP-algorithm and a strong-trip based model. The model is based on a set of nondominated trips that are enumerated by a label-setting algorithm in the first phase, while in the second phase the model is decomposed by Benders and solved using BCP. The Pareto front is examined e. g. by Parragh et al. (2009); Zidi et al. (2012); Chevrier et al. (2012); Paquette et al. (2013); Atahran et al. (2014); Hu et al. (2019) or Viana et al. (2019). A comparison of six evolutionary algorithms to solve the multi-objevtive DARP is provided by Guerreiro et al. (2020).

Contribution. In this paper we suggest an event-based formulation of ride-hailing problems which is based on an abstract graph model and, in contrast to other approaches, has the advantage of implicitly encoding vehicle capacities as well as pairing and precedence constraints. The event-based graph model is the basis for two alternative mixed-integer linear programming formulations that can be distinguished by the approach towards handling ride time and time window constraints. The model has the potential to be used in a rolling-horizon strategy for the dynamic version of the DARP, as it can serve to solve small-sized benchmark instances in a short amount of time. In particular, we show that both suggested MILP formulations outperform a standard formulation from the literature. Furthermore, conflicting optimization goals reflecting, e.g., economic efficiency and customer satisfaction are combined into weighted sum objectives. In addition, we distinguish between average case and worst case formulations. The impact of these objective functions is tested using artificial request data for the city of Wuppertal, Germany.

This paper is organized as follows. After a formal introduction of the problem in Section 3, the event-based graph and based on that two variants of an MILP formulation of the static DARP are presentend in Section 4. The models are compared and tested using CPLEX, in Section 5. The paper is concluded with a summary and some ideas for future research in Section 6.

3. Problem Description

The DARP considered in this paper is defined as follows: Let n be the number of users submitting a transport request. Each request (submitted by a user) $i \in R \coloneqq \{1, \ldots, n\}$ specifies a pick-up location i^+ and a drop-off location i^-. The sets of pick-up and drop-off locations are denoted by $P \coloneqq \{1^+, \ldots, n^+\}$ and $D \coloneqq \{1^-, \ldots, n^-\}$, respectively. A homogeneous fleet of vehicles K with capacity Q is situated at the vehicle depot, which is denoted by 0. A number of requested seats $q_i \geq 1$ and a service duration of $s_i \geq 0$ are associated with each request $i \in R$ and we set $q_{i^+} = q_{i^-} \coloneqq q_i$, $s_{i^+} = s_{i^-} \coloneqq s_i$ and $q_0 \coloneqq 0$ as well as $s_0 \coloneqq 0$. The earliest and latest time at which service may start at a request location $j \in P \cup D \cup \{0\}$ is given by e_j and ℓ_j, respectively, so that each request location has an associated time window $[e_j, \ell_j]$. At the same time, the time window $[e_0, \ell_0]$ corresponding to the depot contains information about the overall start of service e_0 and the maximum duration of service $T \coloneqq \ell_0 - e_0$. The maximum ride time corresponding to request $i \in R$ is denoted by L_i. A feasible solution to the DARP consists of at most $|K|$ vehicle routes which start and end at the depot. If a user is served by vehicle $k \in K$, the user's pick-up and drop-off location both have to be contained in this order in vehicle k's route. The vehicle capacity of Q may not be exceeded at any time. The start of service at every location has to be within the time windows. It is possible to reach a location earlier than the start of service and wait. At each location j, a service duration of s_j minutes is needed for users to enter or leave the vehicle. In addition, the acceptable ride time of each user i is bounded from above by L_i. Vehicles have to return to the depot at least T minutes after the time of the overall start of service e_0. We note that this is a common setting for the static DARP, see, e.g., Gschwind and Drexl (2019); Ropke et al. (2007). Moreover, dial-a-ride problems are often characterized by conflicting objectives. In this paper, we focus on

- economic efficiency, e.g., minimization of routing costs, and
- user experience, e.g., minimization of unfulfilled user requests, waiting time and ride time,

and combine these objectives into an overall *weighted sum objective*. Note that this weighted sum objective can be interpreted as a scalarization of a multi-objective model in which all objective functions are equitably considered. Consequently, by variation of the weights every supported efficient solution can be determined as a solution of the weighted sum scalarization, see e.g. Ehrgott (2005). Other options would be to consider a lexicographic multi-objective model or to treat one objective function, e.g., economic efficiency, as objective and to impose constraints on the other to ensure a satisfactory user experience, which corresponds to the ε-constraint scalarization.

Mixed integer linear programming formulations based on these assumptions are presented in the following section.

4. Mixed Integer Linear Programming Formulations

Classical models for DARP are based on a directed graph that can be constructed in a straight-forward way by identifying all pick-up and drop-off locations and the depot by nodes in a complete directed graph, see e.g. Cordeau (2006) or Ropke et al. (2007). In contrast to this, we suggest an *event-based graph* $G = (V, A)$ in which the node set V consists of Q-tuples representing feasible user allocations together with information on the most recent pickup or drop-off. As will be explained later, this modelling approach has the advantage that many of the complicating routing and pairing constraints can be encoded directly in the network structure: vehicle capacity, pairing (i.e. users' pick-up and drop-off locations need to be served by the same vehicle) and precedence constraints (i.e. users' pick-up locations need to be reached before their drop-off locations are reached) are implicitly incorporated in the event-based graph. Moreover, a directed arc $a \in A$ is only introduced between a pair of nodes if the corresponding sequence of events is feasible w.r.t. the corresponding user allocation. Thus, a tour in G is feasible if it does not violate constraints regarding time windows or ride times.

Using the event-based graph, DARP can be modeled as a variant of a minimum cost flow problem with unit flows and with additional constraints. In particular, we consider circulation flows in G and identify the depot with the source and the sink of a minimum cost flow problem. Each dicycle flow in G then represents one vehicle's tour.

4.1. Event-based graph model

The event-based MILP formulations presented in this paper are motivated by the work of Bertsimas et al. (2019) who propose an optimization framework for taxi routing, where only one passenger is transported at a time. Their algorithm can handle more than $25,000$ users per hour. Bertsimas et al. (2019) propose a graph-based formulation in which an arc (i, j) represents the decision to serve passenger j directly after dropping off passenger i.

The allocation of users to a vehicle with capacity Q can be written as a Q-tuple. Unassigned user slots (i.e., empty seats) are indicated by zeros. If, for example, $Q = 3$ and two requests, request 1 and request 2 with $q_1 = q_2 = 1$, are assigned to the same vehicle, the user allocation may be, for instance, represented by the tuple $(2, 1, 0)$. Accordingly, an empty vehicle is represented by $(0, 0, 0)$. To additionally incorporate information about the most recent pick-up or drop-off location visited, we write

$$(2^+, 1, 0) \quad \text{or} \quad (2^-, 1, 0),$$

if user 1 is seated and user 2 has just been picked-up or dropped-off, respectively. Note that this encoding implicitly specifies the location of the vehicle since, in this example, user 2 is picked-up at his or her pick-up location 2^+, i.e., the 3-tuple $(2^+, 1, 0)$ can be associated with the location 2^+, and user 2 is dropped-off at his or her drop-off location 2^-, i.e., the 3-tuple $(2^-, 1, 0)$ can be associated with the location 2^-. The node $\mathbf{0} = (0, 0, 0)$ is associated with the depot.

Since all permutations of such a Q-tuple specify the same user allocation, we order the components of the Q-tuple such that the first component contains the information regarding the last pick-up or drop-off stop and the remaining $Q-1$ components are sorted in descending order. Users can only be placed together in a vehicle if the total number of requested seats does not exceed the vehicle capacity. This constraint limits the possible combinations of users in the vehicle and hence the set of possible Q-tuples representing user allocations.

Now DARP can be represented by a directed graph $G = (V, A)$, where the node set V represents events rather than geographical locations. The set of event nodes corresponds to the set of all feasible user allocations. The set of all nodes that represent an event in which a request (or user) $i \in R$ is picked up is called the set of *pick-up nodes* and is given by

$$V_{i^+} \coloneqq \Bigg\{ (v_1, v_2, \ldots, v_Q) \colon v_1 = i^+,\ v_j \in R \cup \{0\} \setminus \{i\}\ \forall j \in \{2, \ldots, Q\},$$
$$\Big(v_j > v_{j+1} \vee v_{j+1} = 0\Big)\ \forall j \in \{2, \ldots, Q-1\},\ \sum_{j=1}^{Q} q_{v_j} \leq Q \Bigg\}.$$

Similarly, the set of *drop-off nodes* corresponds to events where a request (or user) $i \in R$ is dropped off and is given by

$$V_{i^-} \coloneqq \Bigg\{ (v_1, v_2, \ldots, v_Q) \colon v_1 = i^-,\ v_j \in R \cup \{0\} \setminus \{i\}\ \forall j \in \{2, \ldots, Q\},$$
$$\Big(v_j > v_{j+1} \vee v_{j+1} = 0\Big)\ \forall j \in \{2, \ldots, Q-1\},\ \sum_{j=1}^{Q} q_{v_j} \leq Q \Bigg\}.$$

Note that for each request $i \in R$ there is only one pick-up and one drop-off location, but more than one potential pick-up node and drop-off node. Hence, there is a unique mapping of nodes to locations, while a location may be associated to many different nodes. For convenience, we write $V_0 \coloneqq \{\mathbf{0}\}$. The overall set of nodes V is then given by

$$V = V_0 \cup \bigcup_{i=1}^{n} V_{i^+} \cup \bigcup_{i=1}^{n} V_{i^-}.$$

The arc set A of G is defined by the set of possible transits between pairs of event nodes in V. It is composed of six subsets, i.e.,

$$A = \bigcup_{i=1}^{6} A_i,$$

where the subsets A_i, $i = 1, \ldots, 6$ are defined as follows:

- Arcs that describe the transit from a pick-up node in a set V_{i^+}, i.e., from a user i's pick-up location, to a drop-off node in V_{j^-}, i.e., to the drop-off

location of a user j, where $j = i$ is possible, but where j may also be another user from the current passengers in the vehicle:

$$A_1 \coloneqq \Big\{ \big((i^+, v_2, \dots, v_Q), (j^-, w_2, \dots, w_Q) \big) \in V \times V : \\ \{j, w_2, \dots, w_Q\} = \{i, v_2, \dots, v_Q\} \Big\}.$$

Note that such arcs reflect the case that the vehicle travels from a user i's pick-up location to a user j's drop-off location, and all users except user j (if there are any) remain seated.

- Arcs that describe the transit from a pick-up node in a set V_{i^+}, i.e., from a user i's pick-up location, to another pick-up node from a set V_{j^+} with $j \neq i$, i.e., to another user j's pick-up location:

$$A_2 \coloneqq \Big\{ \big((i^+, v_2, \dots, v_{Q-1}, 0), (j^+, w_2, \dots, w_Q) \big) \in V \times V : \\ \{i, v_2, \dots, v_{Q-1}\} = \{w_2, \dots, w_Q\} \Big\}.$$

Arcs in A_2 thus represent the trip of the vehicle from a user i's pick-up location to another user j's pick-up location, where user j additionally enters the vehicle.

- Arcs that describe the transit from a drop-off node in a set V_{i^-}, i.e., from a user i's drop-off location, to a pick-up node in a set V_{j^+}, $j \neq i$, i.e., to another user j's pick-up location:

$$A_3 \coloneqq \Big\{ \big((i^-, v_2, \dots, v_Q), (j^+, v_2, \dots, v_Q) \big) \in V \times V : i \neq j \Big\}.$$

- Arcs that describe the transit from a drop-off node in a set V_{i^-}, i.e., from a user i's drop-off location, to a node in V_{j^-}, $j \neq i$, i.e., to another user j's drop-off location:

$$A_4 \coloneqq \Big\{ \big((i^-, v_2, \dots, v_Q), (j^-, w_2, \dots, w_{Q-1}, 0) \big) \in V \times V : \\ \{v_2, \dots, v_Q\} = \{j, w_2, \dots, w_{Q-1}\} \Big\}.$$

Thus, the arcs in A_4 reflect the case that a vehicle travels from a user i's drop-off location to a user j's drop-off location, and all users except user i remain in the vehicle until user j is dropped-off. In particular, user j must already be in the vehicle when user i is dropped-off.

- Arcs that describe the transit from a drop-off node in a set V_{i^-}, i.e., from a user i's drop-off location, to the depot:

$$A_5 \coloneqq \Big\{ \big((i^-, 0, \dots, 0), (0, \dots, 0) \big) \in V \times V \Big\}.$$

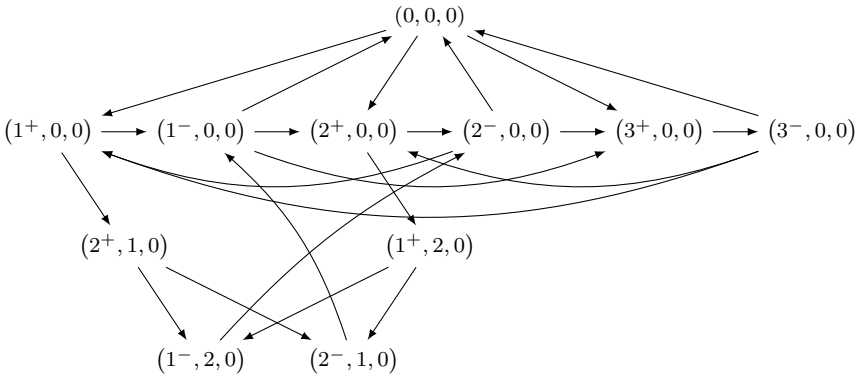

Figure 1: Graph representation of an example with three users.

- Arcs that describe the transit from the depot to a pick-up node in a set V_{i^+}, i.e., to a user i's pick-up location:

$$A_6 \coloneqq \Big\{\big((0,\dots,0),\big(i^+,0,\dots,0\big)\big) \in V \times V\Big\}.$$

Example 1. We give an example of the event-based graph $G = (V, A)$ with three users and vehicle capacity $Q = 3$. Let $R = \{1, 2, 3\}$, $q_1 = q_2 = 2$ and $q_3 = 3$. By the above definitions we obtain the graph illustrated in Figure 1. Note that there are no nodes $v \in V$ that simultaneously contain users 1 (i.e., 1^+ or 1^-) and 3 (i.e., 3^+ or 3^-) as the total number of requested seats specified by these users exceeds the vehicle capacity. Similary, the seats requested by users 2 and 3 together exceed the vehicle capacity of three. Two feasible tours for a vehicle in G are given, for example, by the dicycles

$$C_1 = \Big\{\big((0,0,0),(1^+,0,0)\big),\ \big((1^+,0,0),(2^+,1,0)\big),\ \big((2^+,1,0),(2^-,1,0)\big),\ \big((2^-,1,0),(1^-,0,0)\big),\ \big((1^-,0,0),(0,0,0)\big)\Big\}$$

and

$$C_2 = \Big\{\big((0,0,0),(3^+,0,0)\big),\ \big((3^+,0,0),(3^-,0,0)\big),\ \big((3^-,0,0),(0,0,0)\big)\Big\}.$$

In order to evaluate the complexity of this event-based graph repsentation of DARP, we first evaluate the respective cardinalities of the node set V and of the arc set A. Note that the number of nodes and arcs in the event-based graph model depends on the vehicle capacity, the number of users and the number of requested seats per user. Given Q and n, the number of nodes and arcs is

maximal if all users request only one seat, i.e. if $q_i = 1$ for all $i \in R$. In this case, it is easy to see that

$$|V| = 1 + 2\,n \sum_{j=0}^{Q-1} \binom{n-1}{j}$$

and

$$|A| = 2\,n + n \sum_{j=0}^{Q-1} \binom{n-1}{j}(j+1) + 3\,n\,(n-1) \sum_{j=0}^{Q-2} \binom{n-2}{j} + \frac{n(n-1)\cdot\ldots\cdot(n-Q)}{(Q-1)!},$$

where we use the convention that $\binom{m}{k} \coloneqq 0$ when $k > m$.

From the above formulas, we deduce that the number of nodes is bounded by $\mathcal{O}(n^Q)$ for $n \geq Q$ and the number of arcs is bounded by $\mathcal{O}(n^{Q+1})$ for $n \geq Q+1$. This is in general considerably more than what is obtained when a classical, geometrical DARP model on a complete directed graph is used, which has only $\mathcal{O}(n)$ nodes and $\mathcal{O}(n^2)$ arcs, see e.g. Cordeau (2006); Ropke et al. (2007). However, in practice, ride-hailing services are usually operated by taxis or mini-busses, so that $Q \in \{3, 6\}$. Moreover, the number of nodes and thereby the number of arcs, reduce substantially if we do not consider the "worst-case-scenario" $q_i = 1$ for all $i \in R$, in which all combinations of requests are possible user allocations in the vehicle. Besides that, the event-based formulation has the clear advantage that important constraints like vehicle capacity constraints, pairing constraints and precedence constraints that have to be formulated in classical models are implicitly handled using the event-based graph model, as will be seen in the next section.

4.2. Event-based MILP models

With the above definitions, DARP can be modeled as a minimum cost integer flow problem with additional constraints, where both the source and the sink are represented by the node **0**. We will formulate corresponding MILP models in the subsequent subsections. Therefore, we need the following additional parameters and variables, which are also summarized in Tables 1 and 2.

Since each node in V can be associated with a unique request location $j \in P \cup D \cup \{0\}$, we can associate a routing cost c_a and a travel time t_a with each arc $a = (v, w) \in A$. More precisely, both values c_a and t_a are calculated by evaluating the actual routing cost and travel time from the location associated with v to the location associated with w. We assume that all routing costs and all travel times are nonnegative and satisfy the triangle inequality. Finally, let $\delta^{\text{in}}(v) \coloneqq \{(u, w) \in A \colon w = v\}$ and $\delta^{\text{out}}(v) \coloneqq \{(u, w) \in A \colon u = v\}$ denote the set of incoming arcs of v and the set of outgoing arcs of v, respectively. Moreover, for $a \in A$ let the variable value $x_a = 1$ indicate that arc a is used by a vehicle, and let $x_a = 0$ otherwise. Thus, a vehicle tour is represented by a

Parameter	Description
n	number of transport requests
R	set of transport requests
i^+, i^-	pick-up and drop-off location of request i
P, D	set of pick-up locations/requests, set of drop-off locations
K	fleet of vehicles
Q	vehicle capacity
q_j	load associated with location j
s_j	service duration associated with location j
$[e_j, \ell_j]$	time window associated with location j
T	maximum duration of service
L_i	maximum ride time associated with request i
V	node set
V_{i^+}, V_{i^-}	set of pick-up nodes, set of drop-off nodes corresponding to request i
A	arc set
c_a	routing cost on arc a
t_a	travel time on arc a
t_i	travel time along the shortest path for request i
$\delta^{\text{in}}(v)$, $\delta^{\text{out}}(v)$	incoming arcs, outgoing arcs of node v

Table 1: List of parameters.

Variable	Description
p_i	binary variable indicating if user i is transported or not
B_v	continuous variable indicating the start of service time at node v
x_a	binary variable indicating if arc a is used or not
d_i	excess ride time of user i w.r.t. e_{i^-}
$d_{\max}$	maximum excess ride time

Table 2: List of variables.

sequence of events in a dicycle C in G where $x_a = 1$ for all $a \in C$. A request is matched with a vehicle if the vehicle's tour, i.e., the sequence of events in the corresponding dicycle, contains the event of picking-up and dropping-off of the corresponding user. Since we allow users' requests to be denied, let the variable value $p_i = 1$ indicate that request $i \in R$ is accepted, and let $p_i = 0$ otherwise. Let the variable B_v store the information on the beginning of service time at node $v \in V$, which can be deduced from the variable values x and p. Recall that the beginning of service time has to be within the associated time window of the respective location of node v.

Based on the event-based graph model, we are now ready to formulate our first MILP model for DARP.

4.3. Basic MILP for DARP

In this subsection, we propose an event-based mixed integer linear program for DARP. First, we present a nonlinear mixed integer programming formulation, which is based on the event-based graph model presented in Section 4.1 above. In a second step, this model is transformed into an MILP by a reformulation of time window and ride time constraints, i.e., constraints involving the variables B_v, $v \in V$, using a big-M method.

The DARP can be formulated as the following nonliner mixed integer program:

$$\min \sum_{a \in A} c_a \, x_a \tag{1a}$$

$$s.\,t. \sum_{a \in \delta^{\text{in}}(v)} x_a - \sum_{a \in \delta^{\text{out}}(v)} x_a = 0 \quad \forall v \in V, \tag{1b}$$

$$\sum_{\substack{a \in \delta^{\text{in}}(v) \\ v \in V_{i^+}}} x_a = 1 \quad \forall i \in R, \tag{1c}$$

$$\sum_{a \in \delta^{\text{out}}(\mathbf{0})} x_a \leq |K|, \tag{1d}$$

$$e_j \leq B_v \leq \ell_j \quad \forall j \in P \cup D \cup \{0\},\, v \in V_j \tag{1e}$$

$$(B_w - B_v - s_{i^+}) \sum_{a \in \delta^{\text{in}}(v)} x_a \sum_{a \in \delta^{\text{in}}(w)} x_a \leq L_i \quad \forall i \in R,\, v \in V_{i^+},\, w \in V_{i^-}, \tag{1f}$$

$$B_w \geq (B_v + s_{v_1} + t_{(v,w)})\, x_{(v,w)} \quad \forall (v,w) \in A, \tag{1g}$$

$$x_a \in \{0,1\} \quad \forall a \in A, \tag{1h}$$

$$B_v \geq 0 \quad \forall v \in V. \tag{1i}$$

The objective function (1a) minimizes the total routing cost. While constraints (1b) are flow conservation constraints, it is ensured by constraints (1c) that each request $i \in R$ is accepted and that exactly one node of all nodes which contain the request's pick-up location is reached by exactly one vehicle. Together with constraints (1b) and (1c), the number of feasible dicycles in G is bounded from above by the number of vehicles in constraints (1d). For all nodes in V the start of service has to take place within the time window corresponding to the associated location of the node, which is handled by constraints (1e). An upper bound on the ride time is ensured by constraints (1f). Note that we only impose a bound on the variables B_w, B_v, if both $v \in V_{i^+}$ and $w \in V_{i^-}$ are in fact the pick-up and drop-off nodes that are used to serve request i. Finally, constraints (1g) define the difference in time needed to travel from one node to another. Vehicle capacity, pairing and precedence constraints are ensured by the structure of the underlying network.

This formulation is nonlinear due to constraints (1f) and (1g). In the following MILP these constraints are substitued by a linearized reformulation:

Model I.

$$\min \sum_{a \in A} c_a \, x_a \tag{2a}$$

s. t. constraints (1b) – (1e)

$$B_w - B_v - s_{i^+} \leq L_i + M_i \left(1 - \sum_{a \in \delta^{in}(v)} x_a + 1 - \sum_{a \in \delta^{in}(w)} x_a\right) \quad \forall i \in R,\, v \in V_{i^+}\, w \in V_{i^-}, \tag{2b}$$

$$B_w \geq B_v + s_{v_1} + t_{(v,w)} - \tilde{M}_{v,w} \, (1 - x_{(v,w)}) \quad \forall (v,w) \in A, \tag{2c}$$

$$x_a \in \{0,1\} \quad \forall a \in A, \tag{2d}$$

$$B_v \geq 0 \quad \forall v \in V, \tag{2e}$$

where $M_i \geq \ell_{i^-} - e_{i^+} - L_i - s_{i^+}$ *and* $\tilde{M}_{v,w} \geq \ell_{v_1} - e_{w_1} + s_{v_1} + t_{(v,w)}$ *are sufficiently large constants.*

To include the option to deny requests in DARP, variables $p_i \in \{0,1\}$, $i \in R$ have to be added to Model I and constraints (1c) have to be changed to

$$\sum_{\substack{a \in \delta^{\text{in}}(v) \\ v \in V_{i^+}}} x_a = p_i \qquad \forall i \in R. \tag{3}$$

Hence, if a user is not picked-up (i.e., if $p_i = 0$), then none of the nodes which contain his or her pick-up location are traversed by any vehicle. Note that in this case a reasonable objective function (see Section 4.5) has to penalize the denial of user requests since otherwise an optimal solution is given by $p = 0$, $x = 0$ and $B_v = e_{v_1}$ for all $v \in V$. In the computational experiments in Section 5 we consider both cases, i.e., the scenario that all users have to be served and the scenario that some requests may be denied.

Assuming $q_i = 1$ for all requests $i \in R$ and $n \geq Q + 1$ the total number of variables in Model I can be bounded by $\mathcal{O}(n^{Q+1})$ with $\mathcal{O}(n^{2Q-1})$ constraints, of which $\mathcal{O}(n^{2Q-1})$ constraints are ride time constraints (2b). If $q_i \in \{2, \ldots, Q\}$ for some requests, the number of variables and constraints decreases.

In each of the ride time constraints in Model I, the sums $\sum_{a \in \delta^{\text{in}}(v)} x_a$ and $\sum_{a \in \delta^{\text{in}}(w)} x_a$ are evaluated. This is computationally expensive, as will be demonstrated in the computational tests carried out in Section 5. By taking advantage of the relationship between the pick-up and drop-off time windows associated with request i, we show in the following how constraints (2b) can be reformulated without using big-M constraints, resulting in a second MILP formulation referred to as Model II.

4.4. Reformulation of time-related constraints

The MILP model presented in this subsection differs from the previous model, Model I, in the formulation of time windows and ride time constraints.

In Model I, the ride time constraints are modeled as big-M constraints that are used to deactivate the respective constraints for pick-up and drop-off nodes that are not contained in a vehicle's tour. By reformulating the time window constraints and using the relationship between earliest pick-up and latest drop-off times, the numerically unfavorable big-M constraints can be replaced by simpler constraints in the following model (Model II). This model is faster to solve which is verified by the numerical experiments presented in Section 5.

In this model, the ride time constraints, which ensure that a user does not spend more than L_i minutes in the vehicle, are given by

$$B_w - B_v - s_{i^+} \leq L_i \qquad \forall i \in R, \forall v \in V_{i^+}, \forall w \in V_{i^-}. \tag{4}$$

To show that these constraints, together with a reformulation of constraints (1e), reflect the modeling assumptions, we first observe that in general applications of DARP, as described for example in Cordeau (2006), users often formulate *inbound* requests and *outbound* requests. In the first case, users specify a desired departure time from the origin, while in the case of an outbound request, users specify a desired arrival time at the destination. In both cases a time window is constructed around the desired time, so that we end up with a pick-up time window for inbound requests and a drop-off time window for outbound requests. Now the remaining time window is constructed as follows (based on Cordeau (2006)): For an inbound request the drop-off time window is given by the bounds

$$e_{i^-} = e_{i^+} + s_{i^+} + t_i \qquad \text{and} \qquad \ell_{i^-} = \ell_{i^+} + s_{i^+} + L_i, \tag{5}$$

where t_i denotes the direct travel time $t_{(v,w)}$ from a node $v \in V_{i^+}$ to a node $w \in V_{i^-}$, i.e. $t_i = t_{(v,w)}$ with $v = (i^+, 0, \ldots, 0)$ and $w = (i^-, 0, \ldots, 0)$. Similarly, for an outbound request the pick-up time window is defined by

$$e_{i^+} = e_{i^-} - L_i - s_{i^+} \qquad \text{and} \qquad \ell_{i^+} = \ell_{i^-} - t_i - s_{i^+}. \tag{6}$$

Secondly, we define the notion of an *active* node $v \in V$: We say that a node $v \in V$ is *active* if at least one of its incoming arcs is part of a dicycle flow, i.e., if

$$\sum_{a \in \delta^{\text{in}}(v)} x_a = 1.$$

Otherwise, we call v *inactive*. Note that due to constraints (1c), for each request $i \in R$ we have exactly one associated active pick-up node and one associated active drop-off node. In case we include the option of denying user requests, i.e. we use constraints (3) instead of constraints (1c) and add variables $p_i \in \{0, 1\}$, $i \in R$ to the MILP, each request either has exactly one active pick-up and drop-off node (in this case we have $p_i = 1$), or no associated active node at all (this is the case when $p_i = 0$).

Now, if both v and w in constraints (4) are active nodes, inequalities (4) and (2b) coincide. In case both v and w are inactive, the values B_v, B_w can be ignored in an interpretation of an optimal solution, as v and w are not contained

in any of the vehicle tours. Hence, the critical two cases are the cases where one of the nodes is active and the other node is inactive. Let v^{off} and v^{on} denote the inactive and the active node from the set $\{v, w\}$, respectively. Then, we do not want that $B_{v^{\text{off}}}$ influences the value of $B_{v^{\text{on}}}$ in constraints (4):

Case 1: v is active, w is inactive. Resolving (4) for B_v we obtain $B_v \geq B_w - L_i - s_{i^+}$. Now, we do not want to impose any additional constraints on B_v. Thus, we demand $B_w - L_i - s_{i^+} \leq e_{i^+}$. Accordingly, it has to hold that $B_w \leq e_{i^+} + L_i + s_{i^+}$. Recall that here we assume w to be inactive. If w is active, $B_w \leq \ell_{i^-}$ needs to hold. Putting these restrictions together for an inbound request, we get

$$\begin{aligned} B_w &\leq e_{i^+} + L_i + s_{i^+} + (\ell_{i^-} - (e_{i^+} + L_i + s_{i^+})) \sum_{a \in \delta^{\text{in}}(w)} x_a \\ &= e_{i^+} + L_i + s_{i^+} + (\ell_{i^+} + L_i + s_{i^+} - (e_{i^+} + L_i + s_{i^+})) \sum_{a \in \delta^{\text{in}}(w)} x_a \\ &= e_{i^+} + L_i + s_{i^+} + (\ell_{i^+} - e_{i^+}) \sum_{a \in \delta^{\text{in}}(w)} x_a, \end{aligned} \tag{7}$$

using the reformulation of ℓ_{i^-} from equations (5) in the second step. In the same manner, we use equations (6) to substitute $e_{i^+} = e_{i^-} - L_i - s_{i^+}$ and obtain that

$$B_w \leq e_{i^+} + L_i + s_{i^+} + (\ell_{i^-} - e_{i^-}) \sum_{a \in \delta^{\text{in}}(w)} x_a \tag{8}$$

has to hold for an outbound request. Without loss of generality, we may assume that the length of the time window $[e_{i^+}, \ell_{i^+}]$ constructed around an inbound request has the same length as the time window $[e_{i^-}, \ell_{i^-}]$ constructed around an outbound request, hence the two formulations (7) and (8) coincide.

Case 2: v is inactive, w is active. Resolving (4) for w we obtain $B_w \leq L_i + B_v + s_{i^+}$. In order for the latter inequality to be redundant for B_w, we demand that $L_i + B_v + s_{i^+} \geq \ell_{i^-}$. It follows that $B_v \geq \ell_{i^-} - L_i - s_{i^+}$ has to hold. For an inbound request, using the equivalence from (5), this can be resolved to

$$B_v \geq (\ell_{i^+} + L_i + s_{i^+}) - L_i - s_{i^+} = \ell_{i^+}.$$

Recall that we assumed that v is inactive. In case v is active, the less tighter constraint $B_v \geq e_{i^+}$ has to hold, so that we arrive at

$$B_v \geq e_{i^+} + (\ell_{i^+} - e_{i^+}) \left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right).$$

In a similar fashion, we obtain

$$B_v \geq e_{i^+} + (\ell_{i^-} - e_{i^-}) \left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right),$$

for an outbound request. With the same argumentation as above, we conclude that the two lower bounds on B_v coincide.

As desired, by reformulating the constraints (1e) on the variables B_v, $v \in V$, we obtain a simpler version of the ride time constraints (2b). We put these results together in a second MILP formulation of DARP.

Model II.

$$\min \sum_{a \in A} c_a \, x_a \tag{9a}$$

$$s.\,t. \sum_{a \in \delta^{in}(v)} x_a - \sum_{a \in \delta^{out}(v)} x_a = 0 \quad \forall v \in V, \tag{9b}$$

$$\sum_{\substack{a \in \delta^{in}(v) \\ v \in V_{i^+}}} x_a = 1 \quad \forall i \in R, \tag{9c}$$

$$\sum_{a \in \delta^{out}(\mathbf{0})} x_a \leq |K|, \tag{9d}$$

$$e_0 \leq B_{\mathbf{0}} \leq \ell_0, \tag{9e}$$

$$e_{i^+} + (\ell_{i^+} - e_{i^+}) \left(1 - \sum_{a \in \delta^{in}(v)} x_a\right) \leq B_v \leq \ell_{i^+} \quad \forall i \in R,\, v \in V_{i^+}, \tag{9f}$$

$$e_{i^-} \leq B_v \leq e_{i^+} + L_i + s_{i^+} + (\ell_{i^+} - e_{i^+}) \sum_{a \in \delta^{in}(v)} x_a \quad \forall i \in R,\, v \in V_{i^-}, \tag{9g}$$

$$B_w - B_v - s_{i^+} \leq L_i \quad \forall i \in R,\, v \in V_{i^+},\, w \in V_{i^-}, \tag{9h}$$

$$B_w \geq B_v + s_{v_1} + t_{(v,w)} - \tilde{M}_{v,w} \, (1 - x(v,w)) \quad \forall (v,w) \in A, \tag{9i}$$

$$x_a \in \{0,1\} \quad \forall a \in A, \tag{9j}$$

$$B_v \geq 0 \quad \forall v \in V. \tag{9k}$$

We substitute ride time constraints (2b) for a simpler version (9h). In return we use a more complex version of the time window constraints (9e) – (9g) instead of the short form (1e).

Similar to Model I, variables $p_i \in \{0,1\}$, $i \in R$ may be added to Model II and constraints (9c) may be substituted by constraints (3) to include the option of denying user requests. In this case, the objective function should be modified to contain a term penalizing the denial of requests.

As there are $\mathcal{O}(n^{2Q-1})$ ride time constraints, for each of which two sums $\sum_{a \in \delta^{\text{in}}(v)} x_a$ have to be evaluated in the longer version (2b), but only $\mathcal{O}(n^Q)$ time window constraints, for each of which one sum of the above form has to be evaluated in the longer version (9e) - (9g), we obtain a more efficient new MILP formulation. Similar to Model I, for $n \geq Q+1$ there are at most $\mathcal{O}(n^{Q+1})$ variables and at most $\mathcal{O}(n^{2Q-1})$ constraints, of which $\mathcal{O}(n^{2Q-1})$ constraints are ride time constraints (9h).

4.5. Objective functions

In most of the research on DARP only one objective is used, which is often the minimization of total routing costs. An excellent overview is given by Ho et al. (2018). Other popular objectives are, for example, the minimization of total route duration, number of vehicles used, users' waiting time, drivers' working hours, or deviation from the desired pick-up and drop-off times.

In this paper, we focus on three prevalent (and possibly conflicting) criteria, namely the total routing cost, the total number of unanswered requests and the total excess ride time or the maximum excess ride time, and combine these three criteria into weighted sum objective functions. The first and probably most important criterion is the total routing cost, which can be computed as

$$f_c(x) \coloneqq \sum_{a \in A} c_a \, x_a. \tag{10}$$

We refer to f_c as *cost-objective*. The second objective function,

$$f_n(p) \coloneqq n - \sum_{i \in R} p_i,$$

measures the total number of unsanswered requests. The next optimization criterion relates to customer satisfaction: We measure the response time to a service request by assessing a user's total excess ride time, which aims at penalizing overly long travel times as well as possibly delayed pick-up times.

Let the variable $d_i \geq 0$, $i \in R$ measure the difference in time compared to a user's earliest possible arrival time. We refer to d_i as a user's *excess ride time*. Moreover, let the variabe $d_{\max} \geq 0$ measure the *maximum excess ride time*. By introducing constraints

$$d_i \geq B_v - e_{i^-} \qquad \forall i \in R, \forall v \in V_{i^-}, \tag{11}$$

$$d_{\max} \geq d_i \qquad \forall i \in R, \tag{12}$$

we can now minimize the total or average excess ride time, or the maximum excess ride time (i.e., the excess ride time in the worst case), respectively. The total excess ride time is thus given by the *excess-objective*

$$f_e(d) \coloneqq \sum_{i \in R} d_i, \tag{13}$$

while the *maximum-excess-objective* is given by

$$f_{e_{\max}}(d_{\max}) \coloneqq d_{\max}. \tag{14}$$

The discussion above highlights the fact that we have to consider different and generally conflicting objective functions that are relevant when solving DARP. While the cost objective f_c aims at minimizing total travel cost and thus takes the perspective of the service provider, the quality of service which rather

takes a user's perspective is better captured by objective functions like f_n, f_e and $f_{e_{\max}}$.

We approach this technically bi- or multi-objective problem by using a weighted sum approach with fixed weights, i.e., by combining the relevant objective functions into one weighted sum objective. When using the total excess time as quality criterion, we obtain

$$f_{ce}(x, d) \coloneqq \sum_{a \in A} c_a \, x_a + \alpha \sum_{i \in R} d_i \tag{15}$$

which will be referred to as *cost-excess-objective*, and when using the maximum excess time as quality criterion, we get

$$f_{ce_{\max}}(x, d_{\max}) \coloneqq \sum_{a \in A} c_a \, x_a + \beta d_{\max} \tag{16}$$

which will be referred to as *cost-max-excess-objective*. The parameters $\alpha > 0$ and $\beta > 0$ are weighting parameters that can be selected according to the decision maker's preferences. We refer to Ehrgott (2005) for a general introduction into the field of multi-objective optimization.

Last but not least, we consider a form of DARP in which it is allowed to deny certain user requests. Denying requests can be reasonable if accepting them would mean to make large detours and in turn substantially increase ride times or waiting times of other users. This is accomplished by substituting constraints (1c) in Model I or (9c) in Model II, respectively, by constraints (3) and adding variables $p_i \in \{0, 1\}$, $i \in R$ to Model I or II. In this case, the number of accepted requests has to be maximized or, equivalently, the number of unanswered requests has to be minimized. At the same time, routing costs and excess ride time should be as small as possible. The optimization of these opposing criteria is reflected by the *request-cost-excess-objective* given by

$$f_{rce}(x, d, p) \coloneqq \sum_{a \in A} c_a \, x_a + \alpha \sum_{i \in R} d_i + \gamma \left(n - \sum_{i \in R} p_i \right), \tag{17}$$

where $\gamma > 0$ is an additional weighting parameter. While the third part of the objective refers to the number of unanswered requests and is equal to $\gamma \, n$ at maximum (i.e., if all requests are accepted), the values of the total routing costs and of the total excess ride time strongly depend on the underlying network and request data. Note that meaningful choices of the weighting parameters have to reflect this in order to avoid situations where one part of the objective overrides the others. This will be discussed in Section 5. We emphasize at this point that the objective functions f_{ce}, $f_{ce_{\max}}$ and f_{rce} can be interpreted as weighted sum objectives of bi-objective and tri-objective optimization problems, respectively.

5. Numerical Experiments

This section is divided into two parts. In the first subsection, we compare the MILP formulations Model I and II to a standard formulation of DARP from

Inst.	Name of instance
n	Number of users in the respective instances.
Gap	Gap obtained by CPLEX solution (in percent).
CPU	Computational time in seconds reported by CPLEX.
CPU*	Computational time including time needed to set up MILP in JULIA.
Obj.v.	Objective value.
f_c	Total routing costs.
f_e	Total excess ride time.
$f_{e_{\max}}$	Maximum excess ride time.
a.r.	Answered requests (in percent).
ΔCPU*	Change in computational time including time needed to set up MILP in JULIA (in percent).
Δf_c	Change in total routing costs in percent).
Δf_e	Change in total excess ride time (in percent).
$\Delta f_{e_{\max}}$	Change in maximum excess ride time (in percent).
Δ a.r.	Change in answered requests (in percent).

Table 3: List of abbreviations.

Cordeau (2006). The computational performance is evaluated on a set of benchmark test instances. In the second part, we substitute the cost objective function in Model II with different objective functions as introduced in Section 4.5 and analyze the effect with respect to economic efficiency and customer satisfaction. For this purpose, a set of 70 artificial instances from the city of Wuppertal is generated. The computations are carried out on an Intel Core i7-8700 CPU, 3.20 GHz, 32 GB memory using CPLEX 12.10. The MILPs are programmed using JULIA 1.4.2. with the modelling interface JuMP. The time limit for the solution in all tests is set to 7200 seconds. In the following, the computational results are discussed in detail. We use the abbreviations listed in Table 3.

When computing the average CPU times over several runs, and an instance was not solved to optimality within the time limit, then a CPU (or CPU*) time of 7200 seconds is assumed.

5.1. Benchmark data

We compare our MILP models to a standard formulation of DARP from the literature. For this purpose, we use the basic mathematical model of DARP introduced by Cordeau (2006). Note that a tighter 2-index formulation is given by Ropke et al. (2007). However, this comes at the price of an exponentially growing number of constraints. It is thus better suited for a solution within a B&C framework and we did not include it in our comparison.

The model introduced by Cordeau (2006), referred to as C-DARP in the following, is based on a complete directed graph. The node set comprises all pick-up and drop-off locations and two additional nodes 0 and $2n + 1$ for the depot. Thus, the node set is equal to the set $P \cup D \cup \{0, 2n + 1\}$. We use the MILP formulation with a reduced number of variables and constraints, described in Cordeau (2006), which includes aggregated variables B_j (modelling

Instance	Q	n	$\|K\|$	L_i	T
a2-16	3	16	2	30	480
a2-20	3	20	2	30	600
a2-24	3	24	2	30	720
a3-18	3	18	3	30	360
a3-24	3	24	3	30	480
a3-30	3	30	3	30	600
a3-36	3	36	3	30	720
a4-16	3	16	4	30	240
a4-24	3	24	4	30	360
a4-32	3	32	4	30	480
a4-40	3	40	4	30	600

Instance	Q	n	$\|K\|$	L_i	T
b2-16	6	16	2	45	480
b2-20	6	20	2	45	600
b2-24	6	24	2	45	720
b3-18	6	18	3	45	360
b3-24	6	24	3	45	480
b3-30	6	30	3	45	600
b3-36	6	36	3	45	720
b4-16	6	16	4	45	240
b4-24	6	24	4	45	360
b4-32	6	32	4	45	480
b4-40	6	40	4	45	600

Table 4: Characteristics of the benchmark test instances.

	C-DARP				Model I			Model II		
Inst.	Obj.v.	Gap	CPU	CPU*	Obj.v.	CPU	CPU*	Obj.v.	CPU	CPU*
a2-16	294.2		2.11	2.20	294.2	0.96	8.05	294.2	0.34	2.81
a2-20	344.8		19.61	19.75	344.8	3.80	19.60	344.8	1.14	7.04
a2-24	431.1		108.70	108.92	431.1	12.07	55.87	431.1	2.78	15.28
a3-18	300.5		268.18	268.30	300.5	2.23	11.12	300.5	0.75	4.62
a3-24	346.8	15.6	7200	7200	344.8	33.35	77.35	344.8	14.05	26.25
a3-30	498.0	26.0	7200	7200	494.8	40.43	267.43	494.8	10.89	44.89
a3-36	587.8	19.2	7200	7200	–	–	–	583.2	103.55	196.45
a4-16	282.7	12.3	7200	7200	282.7	2.22	7.58	282.7	0.96	7.68
a4-24	375.0	27.1	7200	7200	375.0	12.47	56.57	375.0	3.66	16.46
a4-32	N/A	N/A	7200	7200	485.5	73.99	426.99	485.5	22.18	78.28
a4-40	N/A	N/A	7200	7200	–	–	–	557.7	279.18	469.18

– Set up of MILP not completed within the time limit.
N/A Not applicable. No integer solution found within the time limit.

Table 5: Solution values and computing times for the benchmark test instances 'a' using JULIA.

the beginning of service time) and Q_j (modelling the vehicle load) at every node j except the origin and destination depot. The objective is to minimize the total routing costs. We do not add any additional valid inequalities to either of the MILP formulations in this comparison.

We use the two sets of benchmark instances[7] set a and set b created by the same author to compare C-DARP to Model I and II. The characteristics of the instances are summarized in Table 4. In all test instances we tighten the remaining time windows, i.e. the time windows not given by the pick-up time of inbound requests or by the drop-off time of outbound requests, respectively, as described in Cordeau (2006): The bounds of the missing time windows can be calculated according to equations (5) and (6) stated earlier in Section 4.

[7]The instances are available at `http://neumann.hec.ca/chairedistributique/data/darp/branch-and-cut/`.

	C-DARP				Model I			Model II		
Inst.	Obj.v.	Gap	CPU	CPU*	Obj.v.	CPU	CPU*	Obj.v.	CPU	CPU*
b2-16	309.4		13.97	15.05	309.4	0.36	2.29	309.4	0.19	1.37
b2-20	332.6		8.70	8.84	332.6	0.03	0.39	332.6	0.03	0.31
b2-24	444.7		90.46	90.68	444.7	0.18	1.72	444.7	0.10	1.09
b3-18	301.6		372.63	372.75	301.6	0.08	0.36	301.6	0.08	0.31
b3-24	394.5		1613.31	1613.52	394.5	1.62	11.05	394.5	0.86	4.82
b3-30	531.9	35.1	7200	7200	531.4	1.06	7.98	531.4	0.43	3.62
b3-36	614.5	22.6	7200	7200	603.8	12.29	41.09	603.8	3.02	12.46
b4-16	297.0		804.32	804.42	297.0	0.04	0.10	297.0	0.04	0.96
b4-24	371.4	16.8	7200	7200	371.4	0.56	3.53	371.4	0.30	2.02
b4-32	501.8	28.2	7200	7200	494.8	0.13	1.39	494.8	0.09	1.06
b4-40	N/A	N/A	7200	7200	656.6	11.83	57.83	656.6	4.02	16.72

N/A Not applicable. No integer solution found within the time limit.

Table 6: Solution values and computing times for the benchmark test instances 'b' using JULIA.

A summary of the computational results can be found in Tables 5 and 6. For each of the three considered models C-DARP, Model I and II, the objective value (Obj.v.) of the cost objective, the computational time in seconds (CPU) and the computational time in seconds including the time needed to set up the MILP in JULIA before it is handed over to CPLEX (CPU*) are reported. The last quantity is included because the solve time returned by CPLEX for Model I and II is rather low, but the time needed to set up the MILP is rather high compared to the set up time of C-DARP. This is due to the fact that, as explained in Section 4, in Model I and II the number of variables and constraints is bounded from above by $O(n^{Q+1})$ and $O(n^{2Q-1})$, respectively, while there are at most $O(n^2)$ variables and constraints in C-DARP. For example, in instance a2-16 the lp-file generated from Model I and II has a size of 180 MB and 24 MB, respectively, while for C-DARP its size is 474 KB. This is reflected by the high difference of the values in columns CPU and CPU* for Model I and II. For C-DARP we also report the relative gap (Gap) as some of the instances have not been solved to optimality within the time limit of 7200 seconds. By taking a closer look at Tables 5 and 6, it becomes evident that Model II outperforms Model I in all of the instances with only one exception at instance b4-16. In all of the instances the CPU time needed by CPLEX to solve Model II is smaller than the solution time for Model I. Instances a3-36 and a4-40 are the two largest instances in terms of the number of users and the possible user allocations in the vehicle (note that $q_i = 1$ for all $i \in R$ in the a-instances). For these instances, the MILP for Model I could not be set up within 7200 seconds, so that we had to interrupt the execution of the program. Modelling the same instances with Model II took about 1.5 minutes (instance a3-36) and about 3 minutes (instance a4-40). Moreover, one can see clearly from the results that Model II yields a more efficient formulation than C-DARP: Starting from instance a3-34, the a-instances could not be solved to optimality within two hours (or even no integer solution was found) with C-DARP. The computational time needed to

solve these instances with Model II ranges from 7.68 seconds to less than eight minutes. In general, with Model II the b-instances are easier to solve than the a-instances, as all instances but instances b3-36 and b4-40 have been solved in less than five seconds. This reflects the fact that the size of the MILP decreases tremendously when users request more than one seat, which can be traced back to the fact that the size of the event-based graph decreases in this case. With C-DARP, CPLEX was able to solve more of the b-instances within the time limit than of the a-instances, but C-DARP still performs poorly when compared to Model II.

5.2. Artificial data – City of Wuppertal

Ride-hailing services are usually operated by taxis or mini-buses, whose passenger capacity is often equal to three or six. Thus, in the artificially created instances we consider the case that $Q \in \{3, 6\}$. In the case that $Q = 3$, we assume that each user requests one seat each, while for $Q = 6$ the number of requested seats per user is chosen randomly from $\{1, \ldots, 6\}$. Moreover, the service time s_j associated with location j is set to be equal to the number of requested seats q_j. This is in accordance with the benchmark test instances for DARP created in Cordeau (2006). The instance size is determined by the number of users. For both $Q = 3$ and $Q = 6$ and for each number of users $n \in \{10, 15, 20, 25, 30, 35, 40\}$ we generate a set of 5 instances with n users each. We denote the instances by $Q3.n.m$ and $Q6.n.m$, indicating the m-th instance with n users, $m \in \{1, \ldots, 5\}$. Pick-up and drop-off locations are chosen randomly from a list of streets in Wuppertal, Germany. The taxi depot is chosen to be located next to the main train station in Wuppertal. The cost c_a for an arc a in the event-based network is computed as the length of a shortest path from its tail to its head in an OpenStreetMap network corresponding to the city of Wuppertal. The shortest path is calculated based on OpenStreetMap data using the shortest path method of the Python package NetworkX. Due to slowly moving traffic within the city center, the average travel speed is set to 15 km/h, so that the travel time in minutes is equal to $t_a = 4\, c_a$. Earliest pick-up times are chosen randomly from five-minute intervals within the next 15–60 minutes (we consider inbound requests only) and the time windows are chosen to be equal to 15 minutes, as we assume that users of ride-hailing services in a city, where public transport operates at high frequencies, want to be picked-up without long waiting times. A user i's maximum ride time L_i is set to 1.5 times the ride time for a direct ride from the pick-up to the drop-off location. The maximum duration of service T is set to 150 minutes. A summary of the artificial instances' remaining characteristics can be found in Table 7.

It has been shown in the previous subsection that Model II performs better than Model I. Therefore, we restrict the following tests to Model II. We compare the impact of employing different objective functions from Section 4.5 on the economic efficiency and the customer satisfaction of the final routing solution. The respective objective functions are used in Model II in the place of (9a). Moreover, for the objective function f_{rce} we add variables p_i, $i \in R$ to Model II and replace constraints (9c) by constraints (3). In case of the objective functions

Instances	Q	n	$\|K\|$	Instances	Q	n	$\|K\|$
Q3.10.1-5	3	10	6	Q6.10.1-5	6	10	6
Q3.15.1-5	3	15	7	Q6.15.1-5	6	15	9
Q3.20.1-5	3	20	9	Q6.20.1-5	6	20	11
Q3.25.1-5	3	25	9	Q6.25.1-5	6	25	15
Q3.30.1-5	3	30	11	Q6.30.1-5	6	30	16
Q3.35.1-5	3	35	12	Q6.35.1-5	6	35	18
Q3.40.1-5	3	40	15	Q6.40.1-5	6	40	20

Table 7: Characteristics of the Wuppertal artificial test instances.

f_e, f_{ce} and f_{rce}, we add variables $d_i \geq 0$, $i \in R$ and constraints (11) to Model II. In case of the objective functions involving the users' *maximum excess ride time*, i.e. $f_{e_{\max}}$ and $f_{ce_{\max}}$, we additionally add the variable $d_{\max} \geq 0$ and constraints (12) to Model II.

The weights in the objective functions involving more than one criterion are chosen from a user perspective and based on extensive numerical tests. In the first part of the computational experiments we consider the single-objective functions f_c, f_e and $f_{e_{\max}}$. The weights in f_{ce} and $f_{ce_{\max}}$ are then chosen so that the values of total and maximum excess ride time, respectively, in f_{ce} and $f_{ce_{\max}}$ deviate not more than 2% from the optimal objective values for f_e and $f_{e_{\max}}$. After some preliminary testing the weights are set to $\alpha = 3$ and $\beta = \frac{3n}{5}$, which yields

$$f_{ce} = \sum_{a \in A} c_a \, x_a + 3 \sum_{i \in R} d_i \quad \text{and}$$
$$f_{ce_{\max}} = \sum_{a \in A} c_a \, x_a + \frac{3n}{5} \, d_{\max}.$$

Note that choosing the weighting parameter β as a function of n ensures that not only the first term in $f_{ce_{\max}}$ grows with the number of users.

Some pick-up and drop-off times or locations may force the drivers to make large detours, which may induce significantly increased routing costs or excess ride times. By using the objective function f_{rce} we can measure the impact of denying certain requests on the served users' excess ride time and the total routing costs. The weighting parameter γ in f_{rce} defines the trade-off between the general requirement of answering as many requests as possible on one hand, and the goal of cost and time efficient transport solutions on the other hand. After testing several weights, it turns out that $\gamma = 60$ is a reasonable choice, yielding

$$f_{rce} = \sum_{a \in A} c_a \, x_a + 3 \sum_{i \in R} d_i + 60 \left(n - \sum_{i \in R} p_i \right).$$

In our numerical experiments, on average at most 10% of the requests are rejected when using these weights. Note that several authors using weighted sum objectives base their choice of weights on Jorgensen et al. (2007) (see e.g. Mauri

	f_c				f_e				
n	f_c	f_e	CPU	CPU*	f_c	f_e	Gap	CPU	CPU*
Q3									
10	78.8	128.2	0.17	1.04	88.1	15.3		0.14	0.93
15	109.1	191.3	0.75	5.20	128.8	34.2		0.79	4.85
20	129.3	240.3	7.04	22.36	154.5	55.6		7.39	21.17
25	156.6	288.1	69.79	111.14	172.7	111.6		73.99	111.86
30	184.7	353.3	179.01	279.99	202.7	141.1		180.42	274.20
35	199.0	403.6	1191.93	1403.76	231.3	131.1	6	3606.83	3756.82
40	241.2	498.0	2944.48	3366.15	280.3	144.6	16	5946.63	6066.63
Q6									
10	80.5	92.5	0.02	0.26	86.3	15.2		0.02	0.25
15	119.1	122.4	0.05	0.63	128.7	20.9		0.05	0.61
20	152.4	178.8	0.20	2.00	164.6	48.0		0.23	1.88
25	183.6	228.0	0.88	6.56	211.1	29.9		0.93	6.14
30	222.8	256.5	1.02	7.14	244.6	46.7		1.06	6.39
35	238.0	340.4	45.51	98.10	277.5	44.7		43.80	86.40
40	296.0	360.4	8.65	31.66	324.1	127.5		9.24	30.51

Table 8: Average values on the Q3 and Q6 test instances solved with the objective functions f_c and f_e.

et al. (2009); Kirchler and Calvo (2013)). However, the total routing costs, the total/maximum excess ride time and the number of unanswered requests depend strongly on the test instances and may vary considerably. Since in this section we create a new class of test instances, we refrain from using these predetermined weights.

In Tables 8, 9 and 10 the results are summarized. All figures reported are average values over five instances $Q3.n.m$, $m \in \{1, \ldots, 5\}$ and $Q6.n.m$, $m \in \{1, \ldots, 5\}$, respectively. The tables contain information about the total routing costs, the total excess ride time (and where applicable the maximum excess ride time and the percentage of answered requests) for each number of user requests $n \in \{10, 15, 20, 25, 30, 35, 40\}$ of the instance sets $Q3$ and $Q6$. Furthermore, the relative gap, the CPU time returned by CPLEX and the CPU time including the set up time in JULIA is reported. If no relative gap is shown, all instances have been solved to optimality.

The computational times show that the artificial instances are harder to solve than the benchmark instances. This may be explained by a smaller planning horizon (240–720 minutes for the benchmark instances and 150 minutes for the artificial instances) during which the same number of user requests have to be served, i.e. the ratio of user requests per time is higher. This is also reflected by the number of required vehicles. While there are only two to four vehicles in the benchmark test instances to serve between 16 and 40 user requests, there are 6 to 20 vehicles required in the artificial instances. The computational time needed to solve the Q6-instances deviates strongly from the time needed to solve the Q3-instances. This is in accordance with the results of the benchmark

	f_{ce}						f_{rce}					
n	Obj.v.	f_c	f_e	Gap	CPU	CPU*	Obj.v.	f_c	f_e	a.r. Gap	CPU	CPU*
Q3												
10	133.6	87.8	15.3		0.10	0.73	129.4	86.2	10.4	98	0.09	0.73
15	230.9	128.2	34.2		0.40	3.56	226.1	121.7	22.4	96	0.35	3.48
20	318.9	151.1	56.0		4.07	14.48	277.9	136.0	15.3	92	3.56	13.50
25	506.0	171.1	111.7		38.91	64.27	409.0	152.9	33.4	90	35.92	61.36
30	624.2	199.7	141.5		99.38	154.37	515.5	183.2	58.8	91	88.15	142.85
35	621.8	227.9	131.3	3	3004.61	3102.21	554.6	216.1	56.8	92 1	2016.91	2113.72
40	712.7	277.3	145.1	9	4945.03	5123.23	638.3	260.7	65.9	92 6	3611.54	3823.74
Q6												
10	131.0	85.4	15.2		0.02	0.26	129.7	81.7	8.0	96	0.02	0.26
15	191.0	128.2	20.9		0.05	0.65	189.8	125.6	17.4	99	0.05	0.61
20	308.4	164.2	48.1		0.22	2.04	281.8	154.0	22.6	95	0.18	2.00
25	296.9	206.3	30.2		0.93	6.55	283.9	202.4	19.2	98	0.92	6.57
30	383.7	243.6	46.7		1.04	6.92	359.7	231.8	22.6	97	0.94	6.80
35	410.3	275.3	45.0		43.81	88.24	407.2	266.5	30.9	98	43.23	88.14
40	704.1	318.3	128.6		9.16	31.96	613.6	297.4	53.4	93	8.10	30.94

Table 9: Average values on the Q3 and Q6 test instances solved with the objective function f_{ce} and f_{rce}.

	$f_{e_{\max}}$					$f_{ce_{\max}}$					
n	f_c	f_e	$f_{e_{\max}}$	CPU	CPU*	Obj.v.	f_c	f_e	$f_{e_{\max}}$	CPU	CPU*
Q3											
10	88.5	40.2	7.3	0.08	0.67	130.0	85.4	46.0	7.4	0.10	0.76
15	130.2	64.2	10.0	0.37	3.35	216.5	126.0	77.8	10.1	0.39	3.63
20	157.0	110.6	11.1	3.70	13.51	280.0	145.5	130.7	11.2	3.69	14.13
25	178.1	179.5	12.3	36.09	59.96	354.4	169.4	199.1	12.3	36.34	62.16
30	210.5	216.0	13.3	87.24	137.65	442.7	201.1	223.5	13.4	87.90	143.22
35	237.2	222.5	10.4	457.55	571.89	448.3	228.6	232.8	10.5	562.73	684.52
40	291.5	258.9	11.4	1139.52	1354.48	552.1	279.1	283.2	11.4	1190.24	1443.64
Q6											
10	86.1	27.3	6.1	0.02	0.25	121.6	84.9	26.9	6.1	0.02	0.26
15	130.1	58.6	8.6	0.04	0.59	203.0	125.4	74.0	8.6	0.06	0.63
20	164.8	93.7	9.6	0.21	1.90	276.0	160.3	102.2	9.7	0.21	2.03
25	212.5	70.8	6.9	0.87	6.00	309.6	206.1	83.7	6.9	0.92	6.35
30	249.2	137.0	11.1	1.02	6.25	436.2	236.4	159.6	11.1	1.06	6.79
35	281.4	131.5	9.4	42.73	85.14	459.6	261.5	191.9	9.4	43.18	88.42
40	330.2	252.0	12.6	8.72	29.82	620.0	318.2	272.7	12.6	8.79	31.75

Table 10: Average values on the Q3 and Q6 test instances solved with the objective functions $f_{e_{\max}}$ and $f_{ce_{\max}}$.

	f_e vs. f_c		f_{ce} vs. f_c		f_{ce} vs. f_e		$f_{e_{\max}}$ vs. f_c		$f_{ce_{\max}}$ vs. f_c		$f_{ce_{\max}}$ vs. $f_{e_{\max}}$	
n	Δf_c	Δf_e	Δf_c	ΔCPU*	Δf_e	ΔCPU*	Δf_c	Δf_e	Δf_c	ΔCPU*	$\Delta f_{e_{\max}}$	ΔCPU*
Q3												
10	12	−88	11	−30	0	−22	12	−69	8	−27	2	14
15	18	−82	18	−32	0	−27	19	−66	15	−30	0	8
20	19	−77	17	−35	1	−32	21	−54	12	−37	1	5
25	10	−61	9	−42	0	−43	14	−38	8	−44	0	4
30	10	−60	8	−45	0	−44	14	−39	9	−49	1	4
35	16	−68	15	121	0	−17	19	−45	15	−51	1	20
40	16	−71	15	52	0	−16	21	−48	16	−57	0	7
Avg.	15	−72	13	−1	0	−28	17	−51	12	−42	1	9
Q6												
10	7	−84	6	−1	0	1	7	−71	6	3	0	8
15	8	−83	8	3	0	7	9	−52	5	0	0	7
20	8	−73	8	2	0	9	8	−48	5	2	0	7
25	15	−87	12	0	1	7	16	−69	12	−3	0	6
30	10	−82	9	−3	0	8	12	−47	6	−5	0	9
35	17	−87	16	−10	1	2	18	−61	10	−10	1	4
40	9	−65	8	1	1	5	12	−30	7	0	0	6
Avg.	11	−80	9	−1	0	6	12	−54	7	−2	0	7

Table 11: Comparison of the change in total routing costs, total excess ride time and CPU* (all in percent) on the Q3 and Q6 test instances solved with the objective functions f_c, f_e, f_{ce}, $f_{e_{\max}}$ and $f_{ce_{\max}}$.

instances, where the b-instances are much faster to solve than the a-instances. While the scenario that users may request any number of seats between one and six reflects the conditions under which ride-hailing services operate in reality, a uniform distribution of q_i in the set $\{1, \ldots, 6\}$ is probably not realistic. In future work, this should be tested at real world data. Nevertheless, in the context of a static framework, the results show that Model II can be solved within reasonable time: For up to 30 users, the average solve time over five Q6 instances is at most seven seconds. The solve time is less than one second for $n \in \{10, 15\}$. Thus, Model II can indeed be applied in a rolling horizon approach for medium sized instances.

A comparison of the effects of different objective functions in Model II can be found in Tables 11 and 12. In the second and third column of Table 11 we illustrate the change (in percent) in total routing costs and total excess ride time when replacing f_c by f_e. While the excess ride time decreases by an average of 72% and 80% (Q3- and Q6-instances, respectively), the routing costs only increase by 15% and 11% on average. This shows that by including the criterion of excess ride time in the objective function we can spare users a large amount of unnecessary ride or waiting time. This comes at the cost of higher routing costs. However, even from a service provider's perspective it might be reasonable to accept a rather small loss in order to improve user convenience and to remain competitive. In column six of the same table we demonstrate that the weights chosen in the cost-excess objective f_{ce} indeed

	$f_{ce_{\max}}$ vs. f_{ce}			f_{rce} vs. f_{ce}			
n	Δf_c	Δf_e	ΔCPU*	Δf_c	Δf_e	ΔCPU*	Δ a.r.
Q3							
10	−3	201	4	−2	−32	0	−2
15	−2	127	2	−5	−35	−2	−4
20	−4	133	−2	−10	−73	−7	−8
25	−1	78	−3	−11	−70	−5	−10
30	1	58	−7	−8	−58	−7	−9
35	0	77	−78	−5	−57	−32	−8
40	1	95	−72	−6	−55	−25	−8
Avg.	−1	110	−22	−7	−54	−11	−7
Q6							
10	0	77	4	−4	−47	2	−4
15	−2	254	−3	−2	−17	−6	−1
20	−2	113	−1	−6	−53	−2	−5
25	0	177	−3	−2	−36	0	−2
30	−3	242	−2	−5	−52	−2	−3
35	−5	326	0	−3	−31	0	−2
40	0	112	−1	−7	−58	−3	−7
Avg.	−2	186	−1	−4	−42	−2	−3

Table 12: Comparison of the change in total routing costs, total excess ride time, CPU* and number of answered requests (all in percent) on the Q3 and Q6 test instances solved with the objective functions f_{ce}, $f_{ce_{\max}}$ and f_{rce}.

reflect a user perspective: There is an increase of at most 1% in total excess ride time compared to solely optimizing w.r.t. f_e. Moreover, there is an average increase in routing costs of 13% (Q3-instances) and 9% (Q6-instances) compared to the costs when using routing costs as the only optimization criterion, i.e. when using f_c as the objective function. Comparing f_{ce} to f_c, we observe a decrease in CPU* time for $Q = 3$ and $n \in \{10, 15, 20, 25, 30\}$ ranging from 30% to 45% and resulting in an average decrease of 1% for the Q3-instances. For the Q6-instances the change (in percent) in CPU* time ranges from -10% to 3%. In comparison to f_e we observe an average decrease in CPU* time of 28% for the Q3-instances but an average increase of 6% for the Q6-instances.

Similar results are obtained for the objective functions f_c, $f_{e_{\max}}$ and $f_{ce_{\max}}$, although the average decrease in excess ride time when comparing $f_{e_{\max}}$ to f_c is only 51% (Q3-instances) and 54% (Q6-instances). The meaningful choice of the weighting parameter in $f_{ce_{\max}}$ is demonstrated by the second last column in Table 11.

Since both of the weighted sum objectives f_{ce} and $f_{ce_{\max}}$ improve user convenience and increase routing costs compared to f_c, we evaluate which of the objective functions is more effective in this respect. Columns two to four in Table 12 illustrate that the average computational time over five instances decreases by up to 78% when using $f_{ce_{\max}}$ instead of f_{ce}. For the Q3-instances we

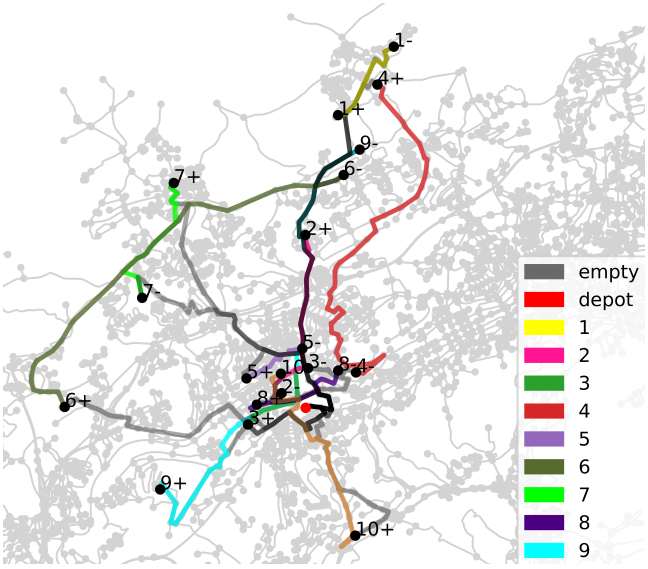

Figure 2: Vehicle routes of instance $Q3n10.3$ solved using the objective function f_{ce}.

observe an average decrease of CPU* time of 22%, for the Q6-instances there is an average decrease of 1%. Despite its computational superiority and the fact that the objective function $f_{ce_{\max}}$ generally improves the user satisfaction of optimal tours, it has some shortcomings. Indeed, if there is one user with a high excess ride time d_i, for instance, because he or she is the last user in a tour to be dropped-off, the time loss of all other users j with $d_j < d_i$ has no impact on the objective function. Therefore, the other users may not be driven to their drop-off points as quickly as possible. This is reflected by the increase in excess ride time using the objective $f_{ce_{\max}}$ as compared to f_{ce}, shown in Table 12: 110% (Q3-instances) and 186% (Q6-instances). The routing costs remain roughly the same; there is an average decrease of 1% (Q3-instances) and 2% (Q6-instances). Due to these shortcomings, we do not consider a tri-criterion weighted sum objective function $f_{rce_{\max}}$, but restrict our attention to f_{rce} in the remaining discussion.

The last four columns of Table 12, f_{rce} vs. f_{ce}, illustrate the decrease in the overall routing costs and excess ride time that is obtained when rejecting some of the requests. This is illustrated at the instance $Q3n10.3$ in Figure 2, showing an optimal tour w.r.t. f_{ce}, and Figure 3 where an optimal tour w.r.t. f_{rce} is shown. In Figure 2 the driver has to make a large detour to transport request 6. When we allow to reject unprofitable requests using f_{rce}, variables $p_i \in \{0, 1\}$, $i \in R$ and constraints (3) instead of f_{ce} and constraints (9c) in Model II, it becomes obvious from Figure 3 that the service provider benefits from a large decrease in routing costs. Table 13 contains the corresponding vehicle routes including pick-up and drop-off times in minutes after the start of service. In the vehicle tours computed using f_{rce}, users 1 and 4 are transported without any

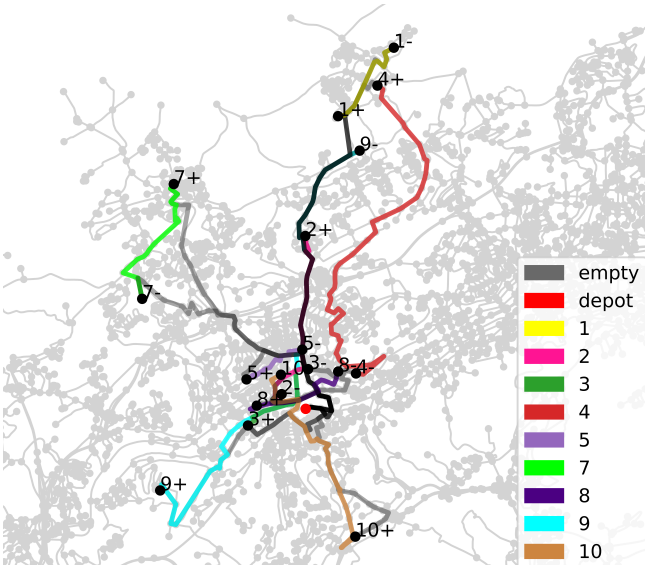

Figure 3: Vehicle routes of instance $Q3n10.3$ solved using the objective function f_{rce}.

		f_{ce}				f_{rce}			
Tour 1	Location	9^+	9^-			9^+	9^-		
	Time[m]	20.0	54.0			20.0	54.0		
Tour 2	Location	3^+	3^-	2^+	2^-	3^+	3^-	2^+	2^-
	Time[m]	15.0	22.3	32.3	45.1	15.0	22.3	32.3	45.1
Tour 3	Location	5^+	5^-	4^+	4^-	4^+	4^-		
	Time[m]	20.0	25.3	45.8	74. 6	35.0	63.8		
Tour 4	Location	8^+	8^-	7^+	7^-	8^+	8^-	7^+	7^-
	Time[m]	20. 0	28.3	48.7	59.9	20. 0	28.3	48.7	59.9
Tour 5	Location	10^+	10^-			10^+	10^-		
	Time[m]	30.0	47.0			30.0	47.0		
Tour 6	Location	6^+	6^-	1^+	1^-	5^+	5^-	1^+	1^-
	Time[m]	20.8	49.1	57.7	65. 0	20.0	25.32	45.0	52.32

Table 13: Vehicle routes (without depot) of instance $Q3n10.3$ solved using the objective functions f_{ce} and f_{rce}.

time loss.

For the Q3- and Q6-instances, on average 7% less and 3% less of the requests are answered in comparison to the results obtained with the objective function f_{ce}. In turn, we observe an average decrease of the total routing costs and the total excess ride time for all instance sizes. While the decrease in routing costs ranges from 2% to 11%, there are huge savings in excess ride time: There is an average decrease of 54% and 42% (Q3- and Q6-instances, respectively.) Furthermore, computational time decreases by 11% (Q3-intances) and 2% (Q6-instances).

While most real world instances of DARPs are much larger, the proposed approach could still prove useful as a subroutine also for realistically-sized instances. Particularly, the extension to the online version of DARP in which requests arrive over time could be a promising application for our models. Indeed, often only very few new requests arrive simultaneously and re-routing of already scheduled users is only acceptable if it decreases their arrival time. Consequently, the number of simultaneous users in such a rolling-horizon version of a dial-a-ride problem is relatively small and could potentially be solved exactly using one of our models. Note that this does in general not lead to a global optimal solution of the offline problem.

6. Conclusions

In this paper we suggest a new perspective on modeling ride-hailing problems. By using an event-based graph representation rather than a geographical model, we show that capacity, pairing and precedence constraints can be handled implicitly. While the resulting MILP formulations generally have more variables as compared to classical models, extensive numerical experiments show that the implicit constraint formulation leads to considerably improved computational times. Indeed, both for benchmark instances from the literature as well as for artificial instances in the city of Wuppertal, problems with up to 40 requests can be solved within a few minutes of computational time, while problems with up to 20 requests can be solved in less than 15 seconds. The new model is thus suitable for an incorporation into dynamic models in a rolling-horizon framework.

Moreover, we analyse the effects of including additional optimization criteria in the model. In addition to the classical cost objective function, we consider the (total or maximum) excess ride time as well as the number of rejected requests as measures for user satisfaction. By combining these criteria into a weighted sum objective function we demonstrate that user satisfaction can be largely improved at only relatively small additional expenses, i.e., overall routing costs.

In the future, we intend to use machine learning techniques to predict requests based on time series data and locate free vehicles accordingly. Furthermore, dynamic vehicle routing algorithms that take into account the traffic volume should be developed such that expected driving times can be computed with high accuracy. The goal is to communicate a reliable estinate of the expected arrival time to the customer when accepting a request.

Acknowledgements

This work was partially supported by the state of North Rhine-Westphalia (Germany) within the project "bergisch.smart.mobility".

References

A. Atahran, C. Lenté, and V. T'kindt. A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. *Journal of Multi-Criteria Decision Analysis*, 21(5-6):279–298, 2014. doi: 10.1002/mcda.1518.

S. Belhaiza. A data driven hybrid heuristic for the dial-a-ride problem with time windows. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, nov 2017. doi: 10.1109/ssci.2017.8285366.

S. Belhaiza. A hybrid adaptive large neighborhood heuristic for a real-life dial-a-ride problem. *Algorithms*, 12(2):39, feb 2019. doi: 10.3390/a12020039.

D. Bertsimas, P. Jaillet, and S. Martin. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, 67(1):143–162, 2019. doi: 10.1287/opre.2018.1763.

C. Bongiovanni, M. Kaspi, and N. Geroliminis. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, 122:436–456, apr 2019. doi: 10.1016/j.trb.2019.03.004.

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

M. Chen, J. Chen, P. Yang, S. Liu, and K. Tang. A heuristic repair method for dial-a-ride problem in intracity logistic based on neighborhood shrinking. *Multimedia Tools and Applications*, apr 2020. doi: 10.1007/s11042-020-08894-7.

R. Chevrier, A. Liefooghe, L. Jourdan, and C. Dhaenens. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport. *Applied Soft Computing*, 12(4):1247–1258, apr 2012. doi: 10.1016/j.asoc.2011.12.014.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006. doi: 10.1287/opre.1060.0283.

J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003. doi: 10.1016/s0191-2615(02)00045-0.

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

C. E. Cortés, M. Matamala, and C. Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711–724, 2010. doi: 10.1016/j.ejor.2009.01.022.

C. Cubillos, E. Urra, and N. Rodríguez. Application of genetic algorithms for the DARPTW problem. *International Journal of Computers Communications & Control*, 4(2):127, 2009. doi: 10.15837/ijccc.2009.2.2420.

J. Desrosiers, Y. Dumas, F. Soumis, S. Taillefer, and D. Villeneuve. An algorithm for mini-clustering in handicapped transport. *Les Cahiers du GERAD, G-91-02, HEC Montréal*, 1991.

P. Detti, F. Papalini, and G. Z. M. de Lara. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, 70:1–14, 2017. doi: 10.1016/j.omega.2016.08.008.

Y. Dumas, J. Desrosiers, and F. Soumis. Large scale multi-vehicle dial-a-ride problems. *Les Cahiers du GERAD, G-89-30, HEC Montréal*, 1989.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2005. ISBN 978-3-540-21398-7. doi: 10.1007/3-540-27659-9. URL `https://doi.org/10.1007/3-540-27659-9`.

T. Garaix, C. Artigues, D. Feillet, and D. Josselin. Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research*, 204(1):62–75, jul 2010. doi: 10.1016/j.ejor.2009.10.002.

T. Gschwind and M. Drexl. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, 53 (2):480–491, 2019. doi: 10.1287/trsc.2018.0837.

T. Gschwind and S. Irnich. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, 49(2): 335–354, 2015. doi: 10.1287/trsc.2014.0531.

P. M. M. Guerreiro, P. J. S. Cardoso, and H. C. L. Fernandes. A comparison of multiple objective algorithms in the context of a dial a ride problem. In *Lecture Notes in Computer Science*, pages 382–396. Springer International Publishing, 2020. doi: 10.1007/978-3-030-50436-6_28.

F. Guerriero, M. E. Bruni, and F. Greco. A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem. *Asia-Pacific Journal of Operational Research*, 30(01):1250046, 2013. doi: 10.1142/s0217595912500467.

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

H. Hosni, J. Naoum-Sawaya, and H. Artail. The shared-taxi problem: Formulation and solution methods. *Transportation Research Part B: Methodological*, 70:303–318, 2014. doi: 10.1016/j.trb.2014.09.011.

T.-Y. Hu and C.-P. Chang. A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost. *Journal of Advanced Transportation*, 49(6):700–723, 2014. doi: 10.1002/atr.1296.

T.-Y. Hu, G.-C. Zheng, and T.-Y. Liao. Multi-objective model for dial-a-ride problems with vehicle speed considerations. *Transportation Research Record: Journal of the Transportation Research Board*, 2673(11):161–171, jun 2019. doi: 10.1177/0361198119848417.

I. Ioachim, J. Desrosiers, Y. Dumas, M. M. Solomon, and D. Villeneuve. A request clustering algorithm for door-to-door handicapped transportation. *Transportation Science*, 29(1):63–78, 1995. doi: 10.1287/trsc.29.1.63.

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

R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society*, 58(10):1321–1331, 2007. doi: 10.1057/palgrave.jors.2602287.

D. Kirchler and R. W. Calvo. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological*, 56:120–135, 2013. doi: 10.1016/j.trb.2013.07.014.

M. Liu, Z. Luo, and A. Lim. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological*, 81:267–288, 2015. doi: 10.1016/j.trb.2015.05.009.

Z. Luo, M. Liu, and A. Lim. A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation. *Transportation Science*, 53(1):113–130, feb 2019. doi: 10.1287/trsc.2017.0772.

M. A. Masmoudi, K. Braekers, M. Masmoudi, and A. Dammak. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & Operations Res.*, 81:1–13, 2017. doi: 10.1016/j.cor.2016.12.008.

G. Mauri, L. Antonio, and N. Lorena. Customers' satisfaction in a dial-a-ride problem. *IEEE Intelligent Transportation Systems Magazine*, 1(3):6–14, 2009. doi: 10.1109/mits.2009.934641.

E. Melachrinoudis, A. B. Ilhan, and H. Min. A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, 34(3):742–759, mar 2007. doi: 10.1016/j.cor.2005.03.024.

Y. Molenbruch, K. Braekers, and A. Caris. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1-2):295–325, 2017. doi: 10.1007/s10479-017-2525-0.

J. Paquette, J.-F. Cordeau, G. Laporte, and M. M. Pascoal. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52:1–16, 2013. doi: 10.1016/j.trb.2013.02.007.

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

S. N. Parragh, K. F. Doerner, R. F. Hartl, and X. Gandibleux. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, 54(4):227–242, dec 2009. doi: 10.1002/net.20335.

Y. Qu and J. F. Bard. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, 49(2):254–270, 2015. doi: 10.1287/trsc.2014.0524.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. doi: 10.1287/trsc.1050.0135.

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

M. Schilde, K. Doerner, and R. Hartl. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1):18–30, oct 2014. doi: 10.1016/j.ejor.2014.03.005.

A. L. S. Souza, J. B. C. Chagas, P. H. V. Penna, and M. J. F. Souza. A hybrid heuristic algorithm for the dial-a-ride problem. In *Variable Neighborhood Search*, pages 53–66. Springer International Publishing, 2020. doi: 10.1007/978-3-030-44932-2_4.

R. J. S. Viana, A. G. Santos, F. V. C. Martins, and E. F. Wanner. Optimization of a demand responsive transport service using multi-objective evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, jul 2019. doi: 10.1145/3319619.3328528.

I. Zidi, K. Mesghouni, K. Zidi, and K. Ghedira. A multi-objective simulated annealing for the multi-criteria dial a ride problem. *Engineering Applications of Artificial Intelligence*, 25(6):1121–1131, sep 2012. doi: 10.1016/j.engappai.2012.03.012.