

A multi-period dial-a-ride problem with driver consistency

Peer-reviewed author version

BRAEKERS, Kris & Kovacs, Attila A. (2016) A multi-period dial-a-ride problem with driver consistency. In: TRANSPORTATION RESEARCH PART B-METHODOLOGICAL, 94, p. 355-377.

DOI: 10.1016/j.trb.2016.09.010

Handle: <http://hdl.handle.net/1942/22527>

A multi-period dial-a-ride problem with driver consistency

Kris Braekers^{a,b,*}, Attila A. Kovacs^c

^a *Research Group Logistics, Hasselt University
Agoralaan Gebouw D, 3590 Diepenbeek, Belgium*

^b *Research Foundation Flanders (FWO)
Egmontstraat 5, 1000 Brussels, Belgium*

^c *CORMSIS Centre of Operational Research,
Management Sciences and Information Systems,
University of Southampton,
Southampton, SO17 1BJ, UK*

Abstract

Dial-a-ride services provide disabled and elderly people with a personalized mode of transportation to preserve their mobility. Typically, several users with different pickup and dropoff locations are transported on a vehicle simultaneously. The focus in Dial-A-Ride Problems (DARPs) is mainly on minimizing routing cost. Service quality has been taken into account in the models by imposing time windows and limiting the maximum ride time of each user. We extend the classical DARP by an additional feature of service quality referred to as driver consistency. Customers of dial-a-ride services are often sensitive to changes in their daily routine. This aspect includes the person who is providing the transportation service, i.e., the driver of the vehicle. Our problem, called the Driver Consistent Dial-A-Ride Problem (DC-DARP), considers driver consistency by bounding the maximum number of different drivers that transport a user over a multi-period planning horizon.

We propose different formulations of the problem and examine their efficiency when applied in a Branch-and-Cut fashion. Additionally, we develop a large neighborhood search algorithm that generates near-optimal solutions in a short amount of time.

Over 1000 instances are generated with close reference to real world scenarios. Extensive computational experiments are conducted in order to assess the quality of the solution approaches and to provide insights into the new problem. Results reveal that the cost of offering driver consistency varies greatly in magnitude. Depending on the instance, the cost of assigning one driver to each user can be up to 27.98% higher compared to a low-cost solution. However, routing cost increases by not more than 5.80% if users are transported by at least two drivers.

Keywords: dial-a-ride problem, multiple periods, driver consistency, Branch-and-Cut, large neighborhood search

*Corresponding author. Tel.: +32.11.269120

Email addresses: kris.braekers@uhasselt.be (Kris Braekers), a.kovacs@soton.ac.uk (Attila A. Kovacs)

1. Introduction

Dial-a-ride services are offered in the context of demand responsive transportation. Users are transported from a specific origin (e.g., from home) to a specific destination (e.g., to a medical facility). Typically, a user has two related transport requests on a single day: an outbound request from home to the desired destination and an inbound request, i.e., the return trip home. Several users with different pickup and dropoff locations may be transported simultaneously to reduce operating costs. Service providers often use several types of vehicles (e.g., cars, minivans, ambulances). A common application is the door-to-door transportation of elderly and disabled people. These people are often unable to use general public transportation. Dial-a-ride services provide a viable and affordable alternative, allowing these people to preserve their mobility. Other applications of dial-a-ride services involve ambulance transports and transportation of people in rural areas with poor public transportation services.

Dial-A-Ride Problems (DARPs) are operational planning problems concerned with designing efficient vehicle routes to fulfill transportation requests of users. Extensive research has been conducted in the past decades to model and solve DARPs efficiently. A literature review is given in Section 2.

The main focus of dial-a-ride service providers has been on minimizing routing cost. Yet, many companies put increasing emphasis on improving service quality in order to increase the satisfaction of their users [1]. Improving service quality often leads to a competitive advantage in markets where a large number of companies compete against each other. Service quality has been considered in DARPs by, e.g., offering time windows and limiting ride times, waiting times, and the number of stops with a user on board. A detailed overview of quality measures is given in Paquette et al. [2].

Most DARP models consider a single day. Therefore, previous research has ignored service quality aspects that arise in a multi-period context. However, a survey of Paquette et al. [1] among users of a dial-a-ride system revealed that the second most important aspect of service quality is being assigned to a familiar driver as often as possible, i.e., driver consistency¹. Users of dial-a-ride services are often sensitive to changes in their daily routine. This includes the person who is providing the transportation service, i.e., the driver of the vehicle.

This paper provides the following contributions: First, for the first time, we study the effect of integrating driver consistency into dial-a-ride problems. In Section 3, we propose a multi-period dial-a-ride problem in which the maximum number of drivers assigned to each user over the planning period is bounded. We consider heterogeneous vehicles and users with different types of requests, e.g., users in wheelchairs and people that have to be transported in a lying position. The problem is referred to as the Driver Consistent Dial-A-Ride Problem (DC-DARP). Second, we propose two mathematical problem formulations in Sections 3.1 and 3.2, respectively. The formulations are strengthened by several valid inequalities and integrated into a Branch-and-Cut (B&C) algorithm that generates optimal solutions for small problem instances (Section 4). Our results provide insights into the advantages of using a

¹The most important aspect is flexibility with respect to reservation changes and booking rides on a short notice.

custom-made B&C algorithm over an off-the-shelf solver. Third, we develop a fast heuristic solution approach based on Large Neighborhood Search (LNS) in Section 5. We put emphasis on designing an algorithm with a simple structure. Therefore, we test several modules and only select the best ones. Fourth, we present an extensive computational study on new benchmark instances with different problem characteristics in Section 6. The efficiency of the problem formulations and the quality of the proposed solution approaches are assessed. Both the exact and heuristic methods provide high-quality results. We analyze the increase in routing cost caused by improved driver consistency and provide recommendations for choosing a proper level of driver consistency in different operating environments. Finally, we draw conclusions and suggest directions for future research in Section 7.

2. Related literature

The proposed problem combines ideas from the DARP literature and the literature on service consistency in vehicle routing problems. The most related papers are reviewed in the next sections.

2.1. Dial-a-ride problems

Many different models and problem variants with a large variety of objective functions and constraints have been proposed to account for the requirements of dial-a-ride service providers. A general problem definition for the DARP is provided by Cordeau and Laporte [3]. This definition considers a limited homogeneous fleet of capacitated vehicles, time windows either on the pickup or the delivery node, maximum ride times limiting the time a user can spend in a vehicle, and a maximum route duration constraint. The problem definition has been adopted by many authors [4, 5, 6, 7, 8, 9, 10, 11, 12]. A three-index formulation for this problem is proposed in Cordeau [6], together with an exact B&C algorithm. Ropke et al. [12] propose two two-index formulations and an improved B&C algorithm. Results show the advantage of using a more compact two-index formulation over the three-index formulation of Cordeau [6]. Heuristic approaches for this problem are based on, e.g., Tabu Search [3], Variable Neighborhood Search (VNS) [9], Large Neighborhood Search (LNS) [7], hybridized Column Generation and LNS [11], and Deterministic Annealing [5].

An extension of the DARP to the heterogeneous case (H-DARP) is proposed in Parragh [10] to account for the fact that service providers often use different types of vehicles and users may have different requirements, especially in the context of transporting elderly and disabled people (e.g., being transported in a wheelchair or on a stretcher). The problem is solved by an exact B&C algorithm and a VNS meta-heuristic. Braekers et al. [5] extend the H-DARP to the multi-depot case. A more compact 2-index formulation is used to obtain better results by an adapted version of the B&C algorithm of Ropke et al. [12]. To the best of our knowledge, Braekers et al. [5] currently provide the best heuristic results for both the standard DARP and its heterogeneous variant.

Other recent contributions focus on rich variants of the DARP which include several real-life aspects (e.g., [13, 14]) and on dynamic pricing strategies (e.g., [15, 16]). Detailed overviews of dial-a-ride problems

are provided by Cordeau and Laporte [17, 4], Parragh et al. [18], and Doerner and Salazar-González [19].

2.2. Service consistency in vehicle routing

In competitive markets, companies often differentiate themselves by providing better customer service than their competitors. This strategy is getting more and more popular in the vehicle routing industry. Companies in, e.g., small package shipping, home health care, grocery home delivery, and demand responsive transportation put large emphasis on service consistency to increase the satisfaction of their customers. Driver consistency is achieved by servicing customers by as few different drivers as possible. Frequent interactions help forming a bond between customer and driver (i.e., the company). Companies focusing on arrival time consistency, service their customers at almost the same time of the day each time they are visited to make arrival times predictable.

Typically, the routing cost increases with a higher level of service consistency. So, the company that provides the highest level of consistency with the least increase in routing cost has the largest advantage over its competitors. This challenge has motivated many researchers to investigate routing problems with service consistency features. Groër et al. [20] formulate the Consistent Vehicle Routing Problem (ConVRP) and develop a record-to-record travel algorithm. Both, arrival time and driver consistency are taken into account. Heuristic solution approaches for the ConVRP are proposed in, e.g., Kovacs et al. [21] and Tarantilis et al. [22]. Customers are visited by a single driver in the ConVRP. Luo et al. [23] and Kovacs et al. [24] examine the effect of allowing more than one driver per customer. Kovacs et al. [25] and Lian et al. [26] investigate the trade-off between routing cost, arrival time consistency, and driver consistency in a multi-objective framework. The first mathematical programming approach for solving routing problems that consider arrival time consistency is presented in Subramanyam and Gounaris [27]. The authors compare three B&C formulations for the multi-period traveling salesman problem in which the difference between the latest and earliest arrival time at each customer is bounded. Service consistency in the context of passenger transportation for people with disabilities is examined in Feillet et al. [28]. The primary focus is on arrival time consistency; the number of drivers per customer is minimized in a post-processing step. Coelho et al. [29] and Coelho and Laporte [30, 31] incorporate service consistency requirements into vendor-managed inventory systems and solve the problem by exact and heuristic algorithms. A detailed overview of vehicle routing problems that consider service consistency is given in Kovacs et al. [32].

3. Problem definition and formulations

In this section, the DC-DARP is defined and two mathematical formulations are proposed. Our problem definition extends the single-day DARP of Cordeau and Laporte [3] to a multi-period problem. We consider driver consistency similar to Kovacs et al. [24], i.e., each user may have multiple requests during the planning horizon, but the number of drivers that can be assigned to a specific user is restricted.

We assume that each user books a set of different types of trips, e.g., trips to medical facilities and day care centers, and social visits to friends and family. Each trip consists of an outbound request, i.e., transport from home to the desired destination, and an inbound request, i.e., the transport back home. A request is characterized by origin, destination, time window at either origin or destination, maximum ride time, and capacity requirement. A new type of trip would be added to a user's set of trips if she required transport to a different location or at a different time of the day.

Each type of trip, and the corresponding transportation requests, may be executed on multiple days of the planning horizon. The requests are assumed to be identical for each of these days (e.g., same time of the day, destination, and capacity requirement). A request that has to be executed on a given day is referred to as active.

Users may even perform multiple trips on a single day. For example, a user may request a transport to a medical facility (and back home) on Monday, Wednesday, and Friday, and a transport to visit a friend on Tuesday and Wednesday (the trips on Wednesday may not overlap). Both types of trips, social visit and medical treatment, have the same characteristics on each day they are demanded. Consistency related constraints are enforced among all requests of a certain user.

A formal description of the DC-DARP by using 4-index decision variables is presented in the next section. A 3-index formulation is presented in Section 3.2.

3.1. 4-index formulation

The following formulation is based on the 3-index formulation of Cordeau [6] for the DARP. Driver consistency and heterogeneous vehicles are incorporated as discussed in Kovacs et al. [24] and Braekers et al. [5], respectively.

Users in set $U = \{1, \dots, |U|\}$ (index u) are serviced over a multi-day planning period $\mathcal{D} = \{1, \dots, |\mathcal{D}|\}$ (index d). Transportation is performed by a set of vehicles $K = \{1, \dots, |K|\}$ (index k). Users require different types of capacity referred to as resource types, e.g., regular seats and wheelchair places. Each vehicle $k \in K$ has a capacity of Q_k^r for resource type $r \in R$. The length of a working day is T_{day} . However, the duration of each vehicle route is restricted to T_{route} .

We assume a permanent vehicle-to-driver assignment, i.e., a driver is always assigned to the same vehicle. The terms drivers and vehicles are used interchangeably.

The DC-DARP is defined on directed graph $G = (N, A)$. The set of nodes $N = \{0, 2n + 1\} \cup P \cup D$ consists of the vehicle depot (i.e., $0, 2n + 1$), the set of pickup nodes $P = \{1, \dots, n\}$, and the set of delivery nodes $D = \{n + 1, \dots, 2n\}$. Delivery node $n + i \in D$ corresponds to pickup node $i \in P$. With each node $i \in N$, we associate a capacity requirement for each resource type q_i^r , a non-negative service duration s_i , a maximum ride time L_i^{max} , and a time window in which the service must start $[e_i, l_i]$. For all $i \in P$, we set $q_i^r = -q_{n+i}^r$ and $s_i = s_{n+i}$. At the depot, we set $q_0^r = q_{2n+1}^r = 0$ and $[e_0, l_0] = [e_{2n+1}, l_{2n+1}] = [0, T_{day}]$. The arc set is defined as $A = \{(i, j) | i, j \in N, i \neq 2n + 1, j \neq 0, i \neq j\}$. With each arc (i, j) , we associate a travel cost c_{ij} and a travel time t_{ij} .

Furthermore, let $\theta(i)$ denote user $u \in U$ corresponding to node $i \in P \cup D$. Binary parameter w_i^d indicates whether the request corresponding to node $i \in P \cup D$ is active on day d ($w_i^d = 1$) or not ($w_i^d = 0$). For each request, we require $w_i^d = w_{n+i}^d$ and $\theta(i) = \theta(n+i)$. Finally, the maximum number of drivers that may be assigned to a certain user is Z (representing the level of consistency).

The following decision variables are used:

$$x_{ij}^{kd} = \begin{cases} 1, & \text{if vehicle } k \text{ traverses arc } (i, j) \text{ on day } d \\ 0, & \text{else} \end{cases} \quad (1)$$

$$y_i^{kd} = \begin{cases} 1, & \text{if node } i \text{ is visited by vehicle } k \text{ on day } d \\ 0, & \text{else} \end{cases} \quad (2)$$

$$z_u^k = \begin{cases} 1, & \text{if vehicle } k \text{ is assigned to user } u \text{ at least once} \\ 0, & \text{else} \end{cases} \quad (3)$$

$$Q_i^{rd} = \text{vehicle load for resource type } r \text{ after visiting node } i \text{ on day } d \quad (4)$$

$$B_i^d = \text{start time of service on day } d \text{ at node } i \in P \cup D \quad (5)$$

$$B_0^{kd} = \text{time vehicle } k \text{ leaves the depot on day } d \quad (6)$$

$$B_{2n+1}^{kd} = \text{return time to depot of vehicle } k \text{ on day } d \quad (7)$$

$$L_i^d = \text{ride time of the request corresponding to pickup node } i \in P \text{ on day } d \quad (8)$$

145

Using the above notation, the DC-DARP may be formulated as:

$$\min \sum_{d \in \mathcal{D}} \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}^{kd} \quad (9)$$

subject to:

$$y_0^{kd} = y_{2n+1}^{kd} = 1 \quad \forall k \in K, d \in \mathcal{D} \quad (10)$$

$$\sum_{i \in P} x_{0i}^{kd} \leq 1 \quad \forall k \in K, d \in \mathcal{D} \quad (11)$$

$$\sum_{i \in D} x_{i, 2n+1}^{kd} \leq 1 \quad \forall k \in K, d \in \mathcal{D} \quad (12)$$

$$\sum_{k \in K} y_i^{kd} = w_i^d \quad \forall i \in P \cup D, d \in \mathcal{D} \quad (13)$$

$$\sum_{j \in N} x_{ji}^{kd} = \sum_{j \in N} x_{ij}^{kd} = y_i^{kd} \quad \forall i \in P \cup D, k \in K, d \in \mathcal{D} \quad (14)$$

$$y_i^{kd} = y_{n+i}^{kd} \quad \forall i \in P, k \in K, d \in \mathcal{D} \quad (15)$$

$$B_j^d \geq (B_0^{kd} + t_{0j})x_{0j}^{kd} \quad \forall j \in P, k \in K, d \in \mathcal{D} \quad (16)$$

$$B_j^d \geq (B_i^d + s_i + t_{ij}) \sum_{k \in K} x_{ij}^{kd} \quad \forall i, j \in P \cup D, d \in \mathcal{D} \quad (17)$$

$$B_{2n+1}^{kd} \geq (B_i^d + s_i + t_{i,2n+1})x_{i,2n+1}^{kd} \quad \forall i \in D, k \in K, d \in \mathcal{D} \quad (18)$$

$$B_{2n+1}^{kd} - B_0^{kd} \leq T_{route} \quad \forall k \in K, d \in \mathcal{D} \quad (19)$$

$$e_i \leq B_i^d \leq l_i \quad \forall i \in P \cup D, d \in \mathcal{D} \quad (20)$$

$$e_i \leq B_i^{kd} \leq l_i \quad \forall i \in \{0, 2n+1\}, k \in K, d \in \mathcal{D} \quad (21)$$

$$L_i^d = B_{n+i}^d - (B_i^d + s_i) \quad \forall i \in P, d \in \mathcal{D} \quad (22)$$

$$t_{i,n+i} \leq L_i^d \leq L_i^{max} \quad \forall i \in P, d \in \mathcal{D} \quad (23)$$

$$Q_j^{rd} \geq (Q_i^{rd} + q_j^r) \sum_{k \in K} x_{ij}^{kd} \quad \forall i, j \in N, r \in R, d \in \mathcal{D} \quad (24)$$

$$q_i^r \leq Q_i^{rd} \leq \sum_{k \in K} Q_k^r y_i^{kd} \quad \forall i \in P, r \in R, d \in \mathcal{D} \quad (25)$$

$$0 \leq Q_i^{rd} \leq \sum_{k \in K} (Q_k^r + q_i^r) y_i^{kd} \quad \forall i \in D, r \in R, d \in \mathcal{D} \quad (26)$$

$$y_i^{kd} \leq z_{\theta(i)}^k \quad \forall i \in P, k \in K, d \in \mathcal{D} \quad (27)$$

$$\sum_{k \in K} z_u^k \leq Z \quad \forall u \in U \quad (28)$$

$$x_{ij}^{kd} \in \{0, 1\} \quad \forall i, j \in N, k \in K, d \in \mathcal{D} \quad (29)$$

$$y_i^{kd} \in \{0, 1\} \quad \forall i \in N, k \in K, d \in \mathcal{D} \quad (30)$$

$$z_u^k \in \{0, 1\} \quad \forall u \in U, k \in K \quad (31)$$

The objective is to minimize the routing cost (9). Constraints (10-12) ensure that each vehicle is assigned to the depot and performs at most one route. Nodes are visited only on days the corresponding requests are active (13). Constraints (14) ensure flow conservation at each node and link the assignment variables (y) with the routing variables (x). Constraints (15) make sure that the pickup and delivery node of a request are served by the same vehicle. These constraints are referred to as pairing constraints. Constraints (16-18) set the start-time-of-service variables (B) and prevent subtours. Maximum route duration and time windows are controlled in (19-21). The maximum ride times of users are defined and bounded in (22) and (23), respectively. Together, (22) and (23) also ensure that a pickup node is visited before its corresponding delivery node (referred to as precedence constraints). Constraints (24) set the load variables (Q), while vehicle capacity is accounted for in (25) for pickup nodes and in (26) for delivery nodes. If a vehicle visits a node, it should be assigned to the corresponding user (27). The number of drivers assigned to each user is restricted in (28). Finally, (29-31) set the domains of the binary decision variables.

The start-time-of-service and load variables, B and Q , are independent of the vehicle performing the service [6]. However, with a heterogeneous fleet, we have to take into account the vehicle-specific capacity

constraints. The right-hand side of constraints (25) and (26) guarantee that the load after servicing a node is bounded by the capacity of the vehicle it has been assigned to. Constraints (16-18) and (24) are nonlinear, but can be linearized using the big-M method. Additionally, the linear form of (24) may be lifted as given in (32) with $M_i \geq \min\{\mathcal{Q}_{max}^r, \mathcal{Q}_{max}^r + q_i^r\}$ and $\mathcal{Q}_{max}^r = \max_{k \in K} \mathcal{Q}_k^r$ [33].

$$\mathcal{Q}_j^{rd} \geq \mathcal{Q}_i^{rd} + q_j^r - M_i(1 - \sum_{k \in K} x_{ij}^{kd}) + (M_i - q_i^r - q_j^r) \sum_{k \in K} x_{ji}^{kd} \quad \forall i, j \in N, r \in R, d \in \mathcal{D} \quad (32)$$

160 3.2. 3-index formulation

The 3-index formulation of the DC-DARP is based on the 2-index *PDPTW2* formulation for the DARP proposed by Ropke et al. [12]. In this formulation, no vehicle index is required on the routing variables. The model requires less variables, but the number of constraints grows exponentially with the size of the input.

165 To take into account vehicle heterogeneity, the node set is extended with both, a dummy pickup node and a dummy delivery node for each vehicle [12, 5]. These nodes have the same physical location as the depot. Each route starts with a dummy pickup node and ends with the corresponding dummy delivery node. Let n^v denote the number of vehicles (i.e., $n^v = |K|$) and n^u the number of user requests; $n = n^v + n^u$. The maximum capacity for resource type $r \in R$ over all vehicles is $\mathcal{Q}_{max}^r = \max_{k \in K} \mathcal{Q}_k^r$.

170 The DC-DARP is defined on directed graph $G = (N, A)$. The node set is $N = \{0, 2n + 1\} \cup P^v \cup P^u \cup D^v \cup D^u$. Nodes 0 and $2n + 1$ represent the depot, $P^v = \{1, \dots, n^v\}$ is the set of dummy pickup nodes, $P^u = \{n^v + 1, \dots, n\}$ is the set of user pickup nodes, $D^v = \{n + 1, \dots, n + n^v\}$ is the set of dummy delivery nodes, and $D^u = \{n + n^v + 1, \dots, 2n\}$ is the set of user delivery nodes. We set $P = P^v \cup P^u$ and $D = D^v \cup D^u$ to simplify notation. The user pickup and delivery nodes $i \in P^u \cup D^u$ have the same attribute values as before (e.g., user, service time, capacity requirement). The attribute values of the dummy nodes are as follows:

$$s_i = s_{n+i} = 0 \quad \forall i \in P^v \quad (33)$$

$$L_i^d = L_{n+i}^d = T_{route} \quad \forall i \in P^v, d \in \mathcal{D} \quad (34)$$

$$[e_i, l_i] = [e_{n+i}, l_{n+i}] = [0, T_{day}] \quad \forall i \in P^v \quad (35)$$

$$w_i^d = w_{n+i}^d = 1 \quad \forall i \in P^v, d \in \mathcal{D} \quad (36)$$

$$q_i^r = -q_{n+i}^r = \mathcal{Q}_{max}^r - \mathcal{Q}_k^r \quad \forall i \in P^v, r \in R, \text{ with } k = i \quad (37)$$

We remove the following arcs from arc set $A = \{(i, j) | i, j \in N, i \neq 2n + 1, j \neq 0, i \neq j\}$ to ensure that each route starts with a dummy pickup node and ends with a dummy delivery node:

$$(0, j) \quad \forall j \in N \setminus P^v \quad (38)$$

$$(i, 2n + 1) \quad \forall i \in N \setminus D^v \quad (39)$$

$$(i, j) \quad \forall i \in P^v, j \in D, j \neq n+i \quad (40)$$

$$(i, j) \quad \forall j \in D^v, i \in P, j \neq n+i \quad (41)$$

Only three types of decision variables are used. The first are the new 3-index binary routing variables. The latter two are the same binary y and z variables as in the 4-index formulation:

$$x_{ij}^d = \begin{cases} 1, & \text{if arc } (i, j) \text{ is traversed on day } d \\ 0, & \text{else} \end{cases} \quad (42)$$

$$y_i^{kd} = \begin{cases} 1, & \text{if node } i \in P \cup D \text{ is visited by vehicle } k \text{ on day } d \\ 0, & \text{else} \end{cases} \quad (43)$$

$$z_u^k = \begin{cases} 1, & \text{if vehicle } k \text{ is assigned to user } u \text{ at least once} \\ 0, & \text{else} \end{cases} \quad (44)$$

Following Ropke et al. [12], we define \mathcal{S} as the set of all subsets of nodes $S \subseteq N$ such that $0 \in S$, $2n+1 \notin S$, and there is at least one node $i \in P$ for which $i \notin S$ and $n+i \in S$. For any subset $S \subseteq P \cup D$, we define $q^r(S) = \sum_{i \in S} q_i^r$. A lower bound on the number of times a vehicle must enter and leave S in order to visit all nodes in the set is $\max\{1, B(S)\}$ with $B(S) = \max_{r \in R} \{\lceil |q^r(S)| / Q_{max}^r \rceil \}$. The set of infeasible paths with respect to time windows, ride times, and the route duration limit is denoted by \mathcal{I} . Finally, set $A(I)$ contains all arcs of path $I \in \mathcal{I}$.

The DC-DARP is formulated as follows:

$$\min \sum_{d \in \mathcal{D}} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}^d \quad (45)$$

subject to:

$$\sum_{k \in K} y_i^{kd} = w_i^d \quad \forall i \in P \cup D, d \in \mathcal{D} \quad (46)$$

$$\sum_{j \in N} x_{ji}^d = \sum_{j \in N} x_{ij}^d = w_i^d \quad \forall i \in P \cup D, d \in \mathcal{D} \quad (47)$$

$$y_i^{kd} = y_{n+i}^{kd} \quad \forall i \in P, k \in K, d \in \mathcal{D} \quad (48)$$

$$y_i^{kd_1} = y_i^{kd_2} \quad \forall i \in P^v, k \in K, d_1, d_2 \in \mathcal{D} \quad (49)$$

$$\sum_{i, j \in S} x_{ij}^d \leq |S| - 2 \quad \forall S \in \mathcal{S}, d \in \mathcal{D} \quad (50)$$

$$\sum_{i, j \in S} x_{ij}^d \leq |S| - \max\{1, B(S)\} \quad \forall S \subseteq N \setminus \{0, 2n+1\}, |S| \geq 2, d \in \mathcal{D} \quad (51)$$

$$\sum_{(i,j) \in A(I)} x_{ij}^d \leq |A(I)| - 1 \quad \forall I \in \mathcal{I}, d \in \mathcal{D} \quad (52)$$

$$y_i^{kd} - y_j^{kd} + x_{ij}^d \leq 1 \quad \forall i, j \in P \cup D, k \in K, d \in \mathcal{D} \quad (53)$$

$$y_j^{kd} - y_i^{kd} + x_{ij}^d \leq 1 \quad \forall i, j \in P \cup D, k \in K, d \in \mathcal{D} \quad (54)$$

$$y_i^{kd} \leq z_{\theta(i)}^k \quad \forall i \in P^u \cup D^u, k \in K, d \in \mathcal{D} \quad (55)$$

$$\sum_{k \in K} z_u^k \leq Z \quad \forall u \in U \quad (56)$$

$$x_{ij}^d \in \{0, 1\} \quad \forall i, j \in N, d \in \mathcal{D} \quad (57)$$

$$y_i^{kd} \in \{0, 1\} \quad \forall i \in N, k \in K, d \in \mathcal{D} \quad (58)$$

$$z_u^k \in \{0, 1\} \quad \forall u \in U, k \in K \quad (59)$$

185 The objective is to minimize the routing cost (45). Constraints (46) make sure that a node is assigned to a vehicle when its corresponding request is active on a given day. Flow conservation is enforced by constraints (47). Constraints (48) ensure that both nodes of a user request are visited by the same vehicle, while constraints (49) ensure that a dummy node is assigned to the same vehicle on each day. Constraints (50), (51), and (52) represent precedence and pairing constraints [34], rounded capacity
190 constraints [35], and infeasible path elimination constraints [36], respectively. The link between the assignment and routing variables is established in constraints (53) and (54): both nodes, i and j , are assigned to the same vehicle if a vehicle travels from i to j on day d . Constraints (55) and (56) are the same as (27) and (28), while (57-59) express the domains of the decision variables.

Constraints (53) and (54) may be lifted by taking subtour elimination and precedence constraints into account. Therefore, constraints (53) can be replaced by constraints (60-62), and constraints (54) by constraints (63-65).

$$y_i^{kd} - y_j^{kd} + x_{ij}^d + x_{ji}^d + x_{n+i,j}^d + x_{n+j,i}^d \leq 1 \quad \forall i, j \in P | i < j, k \in K, d \in \mathcal{D} \quad (60)$$

$$y_i^{kd} - y_j^{kd} + x_{n+i,n+j}^d + x_{n+j,n+i}^d + x_{n+i,j}^d + x_{n+j,i}^d \leq 1 \quad \forall i, j \in P | i < j, k \in K, d \in \mathcal{D} \quad (61)$$

$$y_i^{kd} - y_j^{kd} + x_{i,n+j}^d + x_{n+j,i}^d + x_{n+i,j}^d + x_{j,n+i}^d \leq 1 \quad \forall i, j \in P | i < j, k \in K, d \in \mathcal{D} \quad (62)$$

$$y_j^{kd} - y_i^{kd} + x_{ij}^d + x_{ji}^d + x_{n+i,j}^d + x_{n+j,i}^d \leq 1 \quad \forall i, j \in P | i < j, k \in K, d \in \mathcal{D} \quad (63)$$

$$y_j^{kd} - y_i^{kd} + x_{n+i,n+j}^d + x_{n+j,n+i}^d + x_{n+i,j}^d + x_{n+j,i}^d \leq 1 \quad \forall i, j \in P | i < j, k \in K, d \in \mathcal{D} \quad (64)$$

$$y_j^{kd} - y_i^{kd} + x_{i,n+j}^d + x_{n+j,i}^d + x_{n+i,j}^d + x_{j,n+i}^d \leq 1 \quad \forall i, j \in P | i < j, k \in K, d \in \mathcal{D} \quad (65)$$

For the DARP, the 2-index formulation of Ropke et al. [12] clearly outperforms the 3-index formulation
195 of Cordeau [6]. However, the advantage of the 3-index over the 4-index formulation seems less clear for the DC-DARP. Both models, 3-index and 4-index, require the same assignment variables (y), but the link between these assignment variables and routing variables (x) seems weaker in the 3-index model (constraints (53) and (54) compared to constraints (14)).

4. Exact approaches

Three exact solution approaches for the DC-DARP are considered and compared. The comparison reveals which approach is most preferred for a given problem context. Generating optimal solutions also enables us to assess the quality of the heuristic algorithm presented in Section 5. First, the 4-index model is solved using a state-of-the-art commercial solver (CPLEX). This approach is referred to as *4i-cplex*. Next, both the 4-index model and the 3-index model are solved using a B&C algorithm. These approaches are referred to as *4i-bc* and *3i-bc*, respectively.

In Sections 4.1 to 4.3, we describe methods to reduce the problem size and to improve both models by strengthening the respective formulation. The applied B&C algorithm is discussed in Section 4.4.

4.1. Arc elimination and variable fixing

We apply several preprocessing steps to reduce the problem size. Time windows are tightened and superfluous arcs are eliminated as proposed by Dumas et al. [37] and Cordeau [6] for the single period DARP. For the 4-index model, arcs which are infeasible due to capacity constraints are eliminated for each vehicle separately (as capacity levels may differ between vehicles). Furthermore, for both models, routing and assignment variables related to a node i on a certain day d are only generated if the corresponding request is active on that day ($w_i^d = 1$), and assignment variables y_i^{kd} may be fixed to zero in case vehicle k cannot serve node i due to capacity constraints.

4.2. Branching priority

The DC-DARP may be considered as a two-stage decision process: first, assigning users to vehicles considering driver consistency; second, routing vehicles while ensuring route feasibility. Therefore, it may be beneficial to give branching priority to the assignment variables (y_i^{kd}) over the routing variables (x_{ij}^{kd} or x_{ij}^d) during the Branch-and-Bound (B&B) process.

4.3. Valid inequalities

Additional constraints and valid inequalities may be added to both models in order to strengthen the formulations. These inequalities are redundant in the model, but may reduce the search space. Several valid inequalities for the DC-DARP are discussed in the following paragraphs. They are appended to the problem formulation before starting the solution process, unless stated otherwise. Their effect on solution quality is analyzed in Section 6.2.

Initial pool of inequalities. We add an initial pool of inequalities as proposed in Cordeau [6]. These constraints are simple realizations of the inequalities used to generate cuts in the B&C algorithm (see Section 4.4), i.e., they involve a limited number of variables. In the B&C algorithm, these inequalities are checked one-by-one at each node in the B&B tree, and are added only when a violation is detected.

Incompatibility inequalities. Two requests $i, j \in P$ are denoted as incompatible when they cannot be performed by a single vehicle. This is the case when neither of the following six paths is feasible for any vehicle due to a combination of capacity, time window, and ride time constraints: $\{i, j, n + i, n + j\}$, $\{i, j, n + j, n + i\}$, $\{j, i, n + i, n + j\}$, $\{j, i, n + j, n + i\}$, $\{i, n + i, j, n + j\}$, $\{j, n + j, i, n + i\}$. All arcs between the nodes of incompatible requests are already eliminated by the arc elimination phase discussed in Section 4.1. However, additional incompatibility inequalities may be added as follows. All sets of incompatible requests $S \subseteq P$ with $|S| \geq 2$ are identified by enumeration. Next, for each set $S = \{i_1, i_2, \dots, i_h\}$, inequality (66) ensures that only one such request is assigned to any vehicle k on a given day d .

$$y_{i_1}^{dk} + y_{i_2}^{dk} + \dots + y_{i_h}^{dk} \leq 1 \quad \forall k \in K, d \in \mathcal{D} \quad (66)$$

Symmetry breaking inequalities. Both models proposed for the DC-DARP may be subject to symmetry issues if the fleet is homogeneous or there is only a limited number of vehicle types. Vehicle types are defined such that all vehicles of the same type have the same capacity level Q_k^r for each resource type $r \in R$. Let $K_\omega = \{k_\alpha, k_\beta, \dots, k_\lambda\}$ denote the subset of vehicles belonging to type $\omega \in \Omega$ and $K_\omega^- = K_\omega \setminus \{k_\alpha\}$. We consider two methods to reduce symmetry.

The first method is similar to the variable fixing procedure described in Cordeau [6]. For homogeneous problems, we select the largest set of mutually incompatible requests $S = \{i_1, i_2, \dots, i_h\}$ that are active on a specific day d . Next, these h requests are fixed to the first h vehicles on this day, i.e., $y_{i_1}^{d1} = y_{i_2}^{d2} = \dots = y_{i_h}^{dh} = 1$.

For heterogeneous problems, this method can only be applied when at least one of the resource types can be accommodated by a single type of vehicle only, i.e., a resource type $r \in R$ exists such that requests requiring this resource type can be performed by just a single type of vehicle ω ($\forall k \in K_\omega : Q_k^r > 0, \forall k \notin K_\omega : Q_k^r = 0$). In that case, the largest set of mutually incompatible requests $S = \{i_1, i_2, \dots, i_h\}$ with $q_i^r > 0$ and $w_i^d = 1$ on a specific day d are fixed to the first h vehicles of type ω on that day.

In the second method, we adapt the symmetry breaking constraints of Kovacs et al. [25] for the multi-objective generalized consistent vehicle routing problem. These constraints cannot be applied directly, since in the DC-DARP a single user has multiple requests on a single day (i.e., outbound and inbound). Besides, the constraints have to be added for each type of vehicle separately. Inequalities (67) and (68) represent valid symmetry breaking constraints when $Z = 1$, while inequalities (67) and (69) are also feasible when $Z > 1$.

$$z_u^k \leq \sum_{\substack{d \in \mathcal{D} \\ i \in P | \theta(i)=u}} y_i^{kd} \quad \forall u \in U, k \in K \quad (67)$$

$$z_{u_1}^k \leq \sum_{u_2 \in U | u_2 < u_1} z_{u_2}^{k-1} \quad \forall u_1 \in U, k \in K_\omega^-, \omega \in \Omega \quad (68)$$

$$z_{u_1}^k \leq \sum_{u_2 \in U | u_2 \leq u_1} z_{u_2}^{k-1} \quad \forall u_1 \in U, k \in K_{\omega}^-, \omega \in \Omega \quad (69)$$

Inequalities (67) ensure that z_u^k variables are only equal to 1 if at least a single request of user u is actually assigned to vehicle k . Inequalities (68) allow a user-vehicle assignment (u_1, k) only if vehicle $k - 1$ (of the same type ω) has been assigned to a user u_2 with a smaller index than user u_1 (smaller or equal in case of inequalities (69)).

255

Our results indicate that the first method outperforms the second one. Even better results are obtained when both approaches are combined: first, we fix as many assignment variables to one as possible by using the first method; then, we apply the second method to the remaining vehicles.

Consistency inequalities. When only a single driver per user is allowed over the planning horizon ($Z = 1$), inequalities (70) may be added to both models to indicate that all requests of a certain user should be assigned to the same vehicle over the complete planning horizon.

$$y_i^{d_1, k} = y_j^{d_2, k} \quad \forall i, j \in P | \theta(i) = \theta(j), k \in K, d_1, d_2 \in \mathcal{D} \quad (70)$$

Simple subtour elimination and infeasible path inequalities. Next to the problem specific inequalities presented in the previous paragraphs, both models may also benefit from adding general vehicle routing inequalities. We consider adding simple subtour elimination inequalities for subtours of length two and three, and simple infeasible path inequalities of length three. For the 4-index model, these inequalities are presented in (71), (72), (73) respectively. For the 3-index model, vehicle indices k are removed from these inequalities, and inequality (73) can only be added when the path is infeasible for all vehicles. Infeasible paths of length two are excluded by the arc elimination procedure described in Section 4.1.

$$\sum_{k \in K} (x_{ij}^{dk} + x_{ji}^{dk}) \leq 1 \quad \forall i, j \in N, d \in \mathcal{D} \quad (71)$$

$$\sum_{k \in K} (x_{ij}^{dk} + x_{ji}^{dk} + x_{ih}^{dk} + x_{hi}^{dk} + x_{jh}^{dk} + x_{hj}^{dk}) \leq 2 \quad \forall i, j, h \in N, d \in \mathcal{D} \quad (72)$$

$$x_{ij}^{dk} + x_{jh}^{dk} + x_{ih}^{dk} \leq 1 \quad \forall i, j, h \in N, k \in K, d \in \mathcal{D}, \text{ such that} \\ \text{path } \langle i, j, h \rangle \text{ is infeasible for vehicle } k \quad (73)$$

4.4. Branch-and-cut algorithm

In contrast to the 4-index model, the 3-index model cannot be solved directly using a commercial solver due to the exponential number of constraints. Therefore, a B&C algorithm is used. B&C algorithms combine a B&B procedure with the concept of cutting planes. In a first step, all constraint families of exponential size (i.e., (50), (51), and (52)) are removed from the model and the LP-relaxation is solved. Next, valid cuts are added to the model and the LP-relaxation of the updated model is solved. This procedure is continued until no more valid cuts are found. The resulting solution is optimal if it

260

satisfies the integrality constraints. If the solution is fractional, branching is performed on one of the fractional variables according to the B&B principle and the search continues. Adding valid cuts serves two purposes: ensuring the feasibility of the final solution by excluding solutions that violate the removed constraints, and strengthening the model, i.e., reducing the search space.

We apply the B&C algorithm of Braekers et al. [5] for the H-DARP, which is based on the algorithms of Ropke et al. [12] and Cordeau [6] for the DARP. Seven types of inequalities are used to generate valid cuts: precedence, capacity, strengthened infeasible path, generalized order, subtour elimination, fork, and reachability constraints. The first three are required to ensure feasibility, while the others are used to strengthen the model.

Finally, the 4-index model is also solved using the B&C algorithm for two purposes. First, this allows a comparison between our B&C algorithm and CPLEX on the same model. Second, we can learn which of both models performs best when solved by the B&C algorithm. This is important because the advantage of the 3-index over the 4-index model is less clear for the DC-DARP, as discussed in Section 3.2. To apply the algorithm, the 4-index routing variables are aggregated into 3-index variables similar to those of the 3-index model, i.e., $\hat{x}_{ij}^d = \sum_{k \in K} x_{ij}^{dk}$, and cuts that are violated by these aggregated variables are identified during the search. Although these cuts are not required to ensure feasibility, they may strengthen the model.

5. Heuristic approach

Approximate solutions for the DC-DARP are generated by a Large Neighborhood Search (LNS). LNS is a popular metaheuristic for solving hard combinatorial optimization problems. Starting with an initial solution, the algorithm repeatedly removes requests from the routing plan to reinsert them at better positions. The proposed version of the algorithm is based on the one presented in Kovacs et al. [24] as it outperforms other solution approaches on the ConVRP benchmark instances in terms of average objective value and computation time.

The pseudo code of our LNS is given in Algorithm 1. The initial solution is generated by the repair operator; this solution might violate the driver consistency requirement and it might make use of dummy vehicles in order to assign all requests to a route. The drivers transporting user u are recorded in a list Z_u which are summarized in \mathcal{Z} . The LNS comprises two destroy operators to remove requests (Section 5.1) and one repair operator with several configurations to insert requests (Section 5.2). In each iteration, one destroy operator and one configuration of the repair operator are selected with equal probability (line 3). In line 4, the destroy operator is applied and \mathcal{Z} is updated. As proposed in Kovacs et al. [24], the solution is repaired one day after another in a randomized fashion (lines 5 and 6). The routing plan on day d is denoted by s_d . The repair operator is applied to each day until all active requests on the respective day have been assigned to a driver. Days are repaired in random order because the feasible driver-to-user assignments on a given day depend on the assignments that have been made on other days. For example, if we assign user 1 to driver A on the first day, we have to assign 1 to A on all other days

if users may be transported only by a single driver. The new solution is composed of the daily routing plans (line 8). The decision whether or not to move from the current solution to a new solution is based on a simulated annealing acceptance criterion (lines 9 - 11, Section 5.3). The best solution found so far is replaced, if the new solution is feasible and has a lower objective value (lines 12 - 14). The algorithm stops after a given number of iterations and returns the best solution.

Algorithm 1 LNS

Require: initial solution s , set of drivers assigned to user u , $Z_u \in \mathcal{Z}$

```

1:  $s^{best} = s$ 
2: repeat
3:   select destroy operator  $dest$  and configuration of repair operator  $rep$  randomly
4:    $s' = \text{destroy}(s, dest, \mathcal{Z})$  ▷ update  $\mathcal{Z}$ 
5:   for all  $d \in \mathcal{D}$  do ▷ in random order
6:      $s'_d = \text{repair}(s'_d, rep, \mathcal{Z})$  ▷ consider and update  $\mathcal{Z}$ 
7:   end for
8:    $s' = \bigcup_{d \in \mathcal{D}} s'_d$ 
9:   if  $\text{accept}(s', s)$  then ▷ Section 5.3
10:     $s = s'$ 
11:   end if
12:   if  $f(s') < f(s^{best})$  then
13:     $s^{best} = s'$ 
14:   end if
15: until maximum number of iterations is reached
16: return  $s^{best}$ 

```

We put emphasis on developing a lean but efficient metaheuristic that is widely accepted by practitioners. In addition to the operators used in the final LNS version (described in the next sections), we have tested a greedy based repair operator (Pisinger and Ropke [38], Ropke and Pisinger [39]) and two destroy operators: related removal operator (Shaw [40], Ropke and Pisinger [39]) and cluster removal operator (Ropke and Pisinger [41]). The final operators were selected by running experiments with a small set of randomly chosen instances. Operators that did not contribute to the solution quality were removed. Both versions, the final LNS and the LNS that employs all mentioned operators, produce equally good results when all available instances are considered. The median of the differences in the solution quality is zero, i.e., the final version performs better on one half of the instances while the extended version performs better on the other half. On average, the final version performs 0.1% worse; yet, the computation time is 3% shorter.

5.1. Destroy operators

We employ two destroy operators: random removal and worst removal. Each operator implements a different strategy to remove requests from the solution. Random removal has been applied in, e.g., Pisinger and Ropke [38], Ropke and Pisinger [39], and Ropke and Pisinger [41]. The worst destroy is motivated by Kovacs et al. [24]. In each iteration, the number of requests that will be removed (ν) is selected randomly from the interval $[\min \{0.1 \sum_{i \in P, d \in \mathcal{D}} w_i^d, 30\}, \min \{0.4 \sum_{i \in P, d \in \mathcal{D}} w_i^d, 60\}]$ (Pisinger and Ropke [38]).

The number of vehicles is limited in the DC-DARP. Requests that cannot be assigned to any available vehicle are assigned to dummy vehicles (see Section 5.2). These dummies are deleted with 50% probability before executing a destroy operator. The number of requests that have been removed from the dummy routes is taken into account when determining ν .

Random removal. The random removal operator selects requests randomly and removes them from a randomly chosen day until ν removals have been made. Inbound requests and outbound requests are removed independently from each other. This operator diversifies the search process.

Worst removal. The worst removal operator removes all requests of users that are transported by the most drivers and whose requests incur large routing costs. Each user u is associated with value $\mathcal{C}(u)$ that is composed of the number of different drivers assigned to this user z_u and the average insertion cost per request of this user \bar{c}^u :

$$\mathcal{C}(u) = z_u + \frac{\bar{c}^u}{\sum_{u \in U} \bar{c}^u}. \quad (74)$$

On a given day d , let i' be the predecessor of node i and i'' its successor. The insertion cost of a request associated with pickup node i on day d , c_i^d , is

$$c_i^d = \begin{cases} c_{i'i} + c_{i,i''} + c_{(n+i)',(n+i)} + c_{(n+i),(n+i)''} - c_{i'i''} - c_{(n+i)',(n+i)''} & \text{if } i'' \neq n+i, \\ c_{i'i} + c_{i,(n+i)} + c_{(n+i),(n+i)''} - c_{i',(n+i)''} & \text{if } i'' = n+i. \end{cases} \quad (75)$$

The average insertion cost per request of user u is given by

$$\bar{c}^u = \frac{\sum_{i \in P | \theta(i)=u} \sum_{d \in \mathcal{D}} c_i^d}{\sum_{i \in P | \theta(i)=u} \sum_{d \in \mathcal{D}} w_i^d}. \quad (76)$$

We perturb the calculation of the insertion costs to avoid requests of outlying users to be removed over and over again. This is achieved by adding a randomly chosen number in the interval $[-\eta\bar{c}^u, \eta\bar{c}^u]$ to \bar{c}^u ; η is a parameter defined by the decision maker.

Users are sorted in decreasing order of $\mathcal{C}(u)$ (i.e., in a lexicographic order of driver consistency first and average insertion cost second). User by user, we remove all associated requests from each day they are active until at least ν removals have been made and all assigned users comply with the driver consistency requirement (28).

5.2. Repair operator

The repair operator is based on the regret principle, i.e., it considers the loss that might occur from postponing the insertion of a request to later iterations [42]. For each pickup and delivery pair, we consider all feasible insertion positions and insert the request with the largest difference between its best insertion position and alternative positions in other routes. The operator is applied to each day in the

planning horizon one after another in a randomized fashion. All requests on a certain day are inserted before moving to another day. The user-to-driver assignments on one day affect the feasible assignments on other days. Therefore, we can diversify the search by generating the daily routing plans in a random order.

On a given day d , let c_i^q denote the change in the total travel cost of inserting request i at its cheapest position into its q -cheapest route. Driver consistency is taken into account by penalizing insertion positions that would violate the driver consistency requirements. We add a large penalty ($M = 1000$) to c_i^q if an insertion violates driver consistency. Furthermore, we reduce c_i^q by M if i may not be assigned to additional drivers and the driver of route q is already assigned to serve i on another day. Rewards support the early insertion of requests of users that are already assigned to the maximum number of different drivers.

In each iteration for day d , we select an unassigned request with pickup node i^* and insert it into its cheapest position as follows:

$$i^* := \arg \max_{i \in P | w_i^q = 1} \left\{ \sum_{h=2}^{\min(q, |K|)} (c_i^h - c_i^1) \right\}. \quad (77)$$

Parameter q gives the number of routes that are considered in the repair operator. Expensive or infeasible insertions can be identified and avoided earlier if q is large. We apply four configurations, i.e., $q = \{2, 3, 4, |K|\}$ (Pisinger and Ropke [38], Ropke and Pisinger [39]).

Requests that cannot be assigned without violating the operational constraints (i.e., capacity, time window, maximum ride time, and route duration) are assigned to a dummy vehicle. Dummy vehicles have unlimited capacity but a regular route duration. An additional dummy vehicle is added to K , if a request can be assigned to neither a regular nor a previously added dummy vehicle.

The level of diversification is increased by randomizing the repair phase as suggested in [38, 39]. The insertion cost c_i^q of a request is changed to $c_i^q + y$ where y is a randomly chosen number in the interval $[-\eta c_i^q, \eta c_i^q]$. Parameter η controls the scale of randomization. The repair phase is randomized with a probability of 50%.

We check the feasibility of an insertion with respect to the maximum route duration constraint and the maximum ride time constraints by the eight-step evaluation scheme of Parragh et al. [9], originally introduced in Cordeau et al. [43]. First, we temporarily insert the request into a route. This route is then compressed, i.e., unnecessary waiting times are eliminated by using the concept of forward time slack (Savelsbergh [44]). If the resulting route complies with the route duration constraints, we next check the maximum ride time constraint at each dropoff; if violated, we delay the corresponding pickup time either until the solution becomes feasible, or until delaying a pickup would violate the time window constraints. An insertion position is considered feasible if all but the driver consistency constraints are satisfied.

5.3. Acceptance criterion

Our mechanism for escaping local optima is based on a simulated annealing acceptance criterion [45]. Improving moves are always accepted; moves to solutions with larger evaluation function values are accepted with a certain probability. The evaluation function $g(s)$ is composed of the objective function $f(s)$ and a penalty for violating driver consistency (Kovacs et al. [24]):

$$g(s) = f(s) + (e^{\frac{\max\{0, z^{max} - Z\}}{p_{penalty}}} - 1)i_{LNS}; \quad (78)$$

z^{max} is the maximum number of drivers per user ($z^{max} = \max_{u \in U}\{z_u\}$), and i_{LNS} is the current LNS iteration. Parameter $p_{penalty}$ is set such that the penalty for violating the driver consistency by one driver in the last iteration, i_{LNSmax} , is equal to the objective value of the initial solution, $s^{initial}$. This yields

$$p_{penalty} = \frac{1}{\ln(\frac{f(s^{initial})}{i_{LNSmax}} + 1)}. \quad (79)$$

375 The initial solution might violate the driver consistency requirement and might also exceed the available fleet size. As a consequence, $f(s^{initial})$ could be lower than the objective value of the optimal solution. To account for this issue, we multiply the objective value of infeasible solutions by parameter λ .

A new solution s' replaces the current incumbent s if $g(s') < g(s)$. Otherwise, a move to s is accepted with probability $e^{-(g(s') - g(s))/\hat{t}}$. Parameter \hat{t} is the temperature in the annealing process. Initially, it is set to accept a solution that is $w_{\hat{t}}$ % worse than the initial solution with a probability of 50% [39, 38]:

$$\hat{t} = -\frac{w_{\hat{t}}}{\ln 0.5} f(s^{initial}). \quad (80)$$

The cooling rate is controlled by parameter c , i.e., \hat{t} is replaced by $\hat{t}c$ after each LNS iteration.

380 The best found solution s^{best} is replaced if s' is feasible and $f(s') < f(s^{best})$.

6. Computational experiments

Extensive numerical experiments have been conducted to assess and compare the quality of the proposed solution approaches, and to examine the effect of imposing driver consistency constraints in a dial-a-ride context.

385 Both, exact and heuristic, algorithms are coded in C++. For the exact approaches, the applied solver is CPLEX 12.6.1 (using the Concert Technology). Default settings are applied unless mentioned otherwise. Only a single thread is allowed and the runtime limit is 4h. Experiments with the exact algorithms are run on a Xeon E5-2680v2 CPU at 2.8GHz with 64GB of RAM whereas the LNS is run on an Intel Xeon X5550 computer with 2.67GHz.

| Instance type | Description |
|---------------|---|
| h0/h1 | Heterogeneity (no, yes) |
| t0/t1 | Type of trips per user (1, ≤ 2) |
| f0/f1/f2 | Frequency of trip types (low, medium, high) |
| c0/c1/c2 | Level of clustering (none, only destination, all) |

Table 1: Summary of instance characteristics.

Benchmark instances for the DC-DARP are introduced in Section 6.1, while parameter settings of the different solution approaches are discussed in Section 6.2. A comparison between the exact approaches and a comparison between the exact and heuristic approaches are presented in Sections 6.3 and 6.4, respectively. Finally, the effect of introducing driver consistency is analyzed (Section 6.5).

6.1. Data sets

A new set of benchmark instances is generated as we are not aware of any benchmark data for multi-period DARPs. We propose a new data set rather than adapting instances for the single period (H-)DARP. Existing instances often exhibit characteristics that are undesirable or unrealistic in the DC-DARP, e.g., outbound and inbound requests are not linked, maximum ride times are independent of the direct ride time $t_{i,n+i}$. However, relevant aspects such as vehicle heterogeneity have been modeled similarly as in existing instances. All instances are available online at

<http://alpha.uhasselt.be/kris.braekers>.

432 problem instances with diverse characteristics have been generated in total. Each instance is solved for three levels of driver consistency ($Z = 1, 2, 3$), resulting in 1296 combinations. We consider a five day planning period, with 12 hours per day (i.e., 8 a.m. – 8 p.m.). Origins and destinations are generated in a square region of 20km². The vehicle depot is located in the center of the region. The fleet is sufficiently large to serve all requests. Total route duration is equal to 8 hours and an average driving speed of 40km/h is assumed. Time windows of 15 minutes are imposed on the delivery node for outbound requests and on the pickup node for inbound requests. Maximum user ride time for a request is two times the direct ride time, with a minimum of 20 minutes to ensure sufficient planning flexibility, i.e., $L_i^{max} = \max\{20, 2 \times t_{i,n+i}\}$. Each user has a home location, a capacity requirement for each resource type, and a service time of 5 minutes for regular users and 10 minutes for users being transported in a wheelchair or on a stretcher.

The number of users ranges from 5 to 100 (5 to 50 in steps of 5, plus 75 and 100, i.e., 12 levels in total). We categorize instances according to four characteristics, with two or three levels per characteristic. An overview is given in Table 1. A single instance is generated for each combination of instance characteristics and number of users.

First, homogeneous (h0) and heterogeneous instances (h1) are considered similar to the instances of Parragh [10]. In the homogeneous instances, only a single resource type is available, accompanying persons are prohibited, and all vehicles have the same capacity (i.e., 3 seats). In the heterogeneous instances, we consider two types of vehicles with four types of resources each (i.e., regular, wheelchair,

| New type | Capacity level | User requirements |
|----------|--|---|
| Type A | Staff seats + patient seats + stretchers | Accompanying persons + seated patients + patients on stretchers |
| Type B | Patient seats + stretchers | Seated patients + patients on stretchers |
| Type C | Stretchers | Patients on stretcher |
| Type D | Wheelchair places | Patients in wheelchair |

Table 2: Redefinition of capacities and resource types.

| Instance type | First trip type | | | | | Second trip type | | | | |
|---------------|-----------------|-----|-----|-----|-----|------------------|-----|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| f0 | 0.6 | 0.1 | 0.1 | 0.1 | 0.1 | 0.9 | 0.1 | 0 | 0 | 0 |
| f1 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.7 | 0.3 | 0 | 0 | 0 |
| f2 | 0.1 | 0.1 | 0.1 | 0.1 | 0.6 | 0.5 | 0.5 | 0 | 0 | 0 |

Table 3: Probabilities for number of times a trip type is active during planning horizon.

stretcher, and accompanying person). We use the same vehicle types, probabilities for users to request a resource type, and upgrading conditions as in Parragh [10]. The upgrading conditions are already incorporated in the instances by defining four new types of capacities and resources as presented in Table 2.

Second, instances are distinguished by the number of different types of trips of users. In some instances, users only make a single type of trip (t0), e.g., to a medical facility, while in other instances users may require two types of trips (t1), e.g., to a medical facility and to visit a friend. The probability of assigning a second type of trip to a user is 20%. Each type of trip has a certain destination and a period of time, between 30 minutes and 4 hours, during which the user wishes to be present at this destination. Each type of trip may be demanded on one or more days during the planning horizon, and always results in an outbound transportation request (from home to the destination) and an inbound transportation request (back home).

Third, the number of times a trip is active during the planning horizon varies by three levels (f0, f1, f2). For each level, Table 3 indicates the probabilities for activating a certain number of trips of a given type. These trips are then randomly distributed over the days. For t0-instances, only the left part of Table 3 is relevant. For t1-instances, separate probabilities for the first and second trip type are applied (left and right part of Table 3, respectively). Time windows are generated such that trips cannot overlap in time if a user makes two trips on a single day.

Finally, different geographical distributions of the locations are considered. Locations are distributed randomly in the instances of type c0. For instances of type c1 and c2, locations are clustered based on the method in Cordeau et al. [43]. The number of seed points around which locations are clustered is either two or three (chosen randomly) and ϕ , a constant to control the level of clustering, is equal to 0.8. The locations of the seed points are randomly generated, while ensuring a distance of at least 8 km between them. Instances of type c1 have randomly dispersed home locations, but the destination locations are clustered. Destinations are clustered around the seed point closest to the corresponding home location with a probability of 0.8. The probability of assigning a destination to any other seed point is 0.2. Both, home locations and destination locations, are clustered in the c2-instances. The probability of assigning

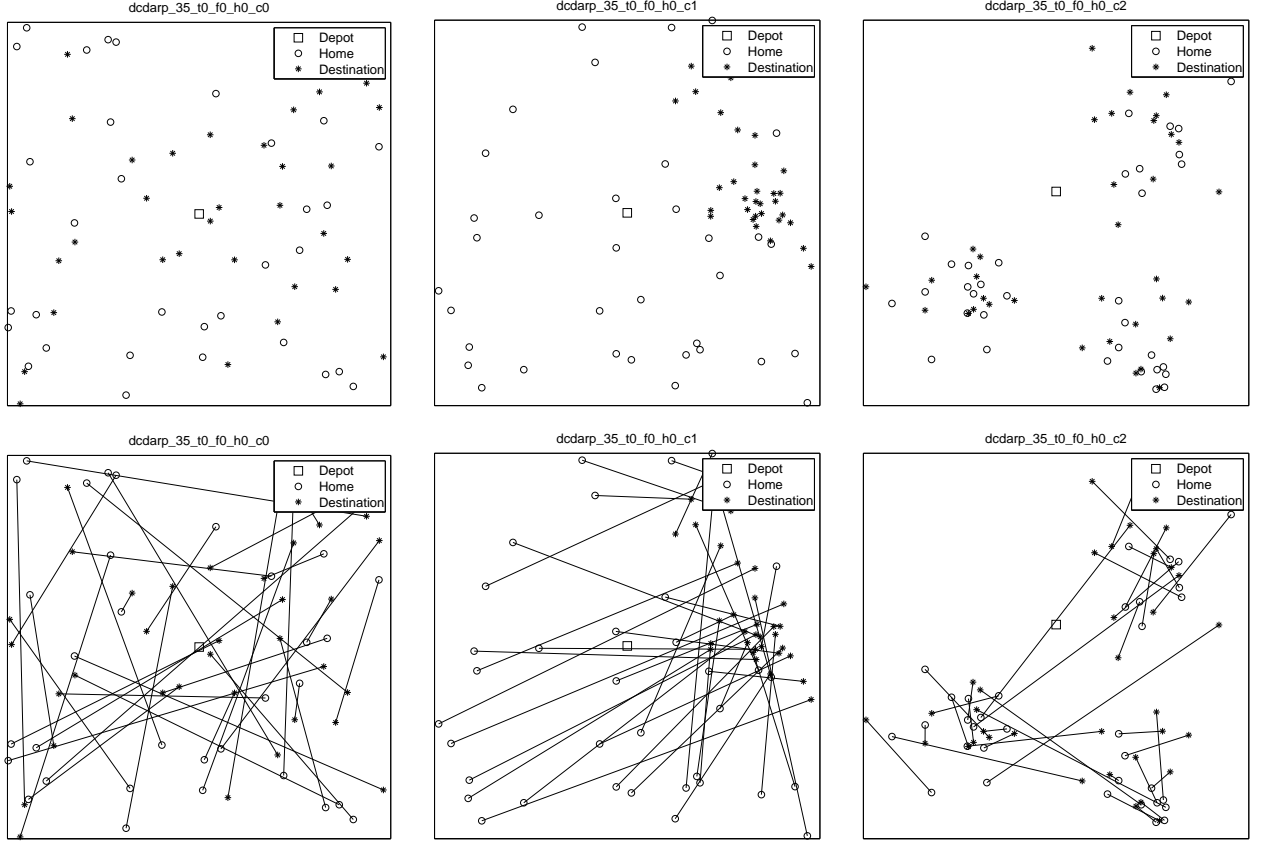


Figure 1: Different levels of clustering. The figures at the top show the pickup and dropoff locations. The figures at the bottom show how the locations are paired.

a destination and the corresponding home location to the same cluster is 0.8. The clustering effect is illustrated in Figure 1 for instance types c0, c1, and c2 with 35 users, respectively.

The number of active requests of a user u during the planning horizon ($\sum_{i \in P | \theta(i)=u} \sum_{d \in \mathcal{D}} w_i^d$) is 6.21 on average. However, the number of active requests per user varies greatly between instances with low frequencies (f0, 4.16 active requests) and those with high frequencies (f2, 8.29 active requests). The total number of active requests per day mainly depends on the number of users. Instances with 5 and 100 users have 6.1 and 125.3 active requests per day, on average, respectively. The largest instance in the set has 888 active requests over the 5-day planning period (177.6 per day on average). An overview of the relation between the number of users and the number of active requests over the 5-day planning period is shown in Figure 2. For each number of users the minimum, average and maximum number of active requests are indicated.

6.2. Parameter settings

The effect of introducing branching priorities (Section 4.2) and valid inequalities (Section 4.3) on the 4-index and 3-index model is analyzed by running different configurations of both models on 18 instances. Each instance is solved for a specific value of $Z = \{1, 2, 3\}$. The instances are selected such that all levels

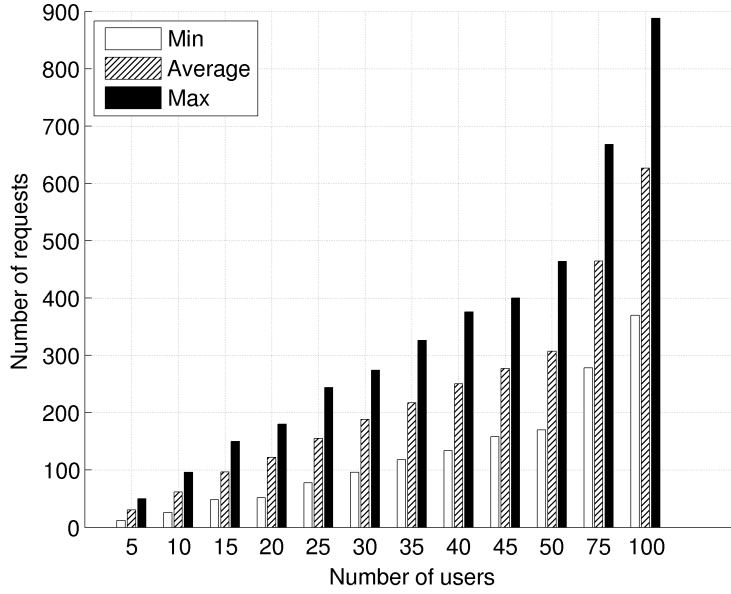


Figure 2: Number of users vs. number of active requests over the 5-day planning period.

of an instance characteristic are equally represented. The same holds for the three levels of Z . The 4-index model is solved using CPLEX, while the 3-index model is solved using the B&C algorithm.

Results of this analysis are given in Table 4. The first line represents the base setting (i.e., without branching priority and valid inequalities). The following eight lines show results when either branching priority or one of the valid inequalities is activated. For each setting, we present the percentage of instances (out of 18) that were solved to optimality within 4 hours, the average gap between the lower bound provided by the respective setting (LB) and the best known upper bound² (UB_{bk}), calculated as $(UB_{bk} - LB)/UB_{bk}$, and the average computation times in minutes.

Introducing branching priority seems to be harmful when applied individually. The efficiency of both formulations decreases in terms of average gap or number of instances solved to optimality. For the 4-index model, the same holds for individually adding simple subtour elimination constraints. Since we use CPLEX as a black box solver, we have no insight into the reason for this. All other valid inequalities improve the results compared to the base setting. The best performance can be achieved by combining different valid inequalities. The last line shows the best results for each model. The best results for the 4-index model are obtained when all inequalities and branching priority are activated. The 3-index model provides the best results without branching priority and with all inequalities enabled except subtour elimination constraints with $|S| = 2$. For both models, efficiency is improved considerably compared to the base setting. The best settings are applied in all further experiments.

For the heuristic approach, the number of LNS iterations is 50000. Table 5 shows the chosen parameter

²The best known upper bound refers to the best solution found while experimenting with exact and heuristic approaches.

| Setting | 4-index model (<i>4i-cplex</i>) | | | 3-index model (<i>3i-bc</i>) | | |
|---|-----------------------------------|------------------------------|------------------|--------------------------------|------------------------------|------------------|
| | <i>Opt</i> (%) | <i>Gap_{avg}</i> (%) | <i>CPU</i> (min) | <i>Opt</i> (%) | <i>Gap_{avg}</i> (%) | <i>CPU</i> (min) |
| Base setting | 33.33 | 6.37 | 163 | 38.89 | 5.10 | 147 |
| Branching priority | 27.78 | 6.50 | 173 | 33.33 | 5.07 | 161 |
| Initial pool of ineq. | 38.39 | 5.86 | 148 | 38.89 | 4.92 | 148 |
| Incompatibility ineq. | 33.33 | 5.96 | 172 | 38.89 | 4.99 | 148 |
| Symmetry breaking ineq. | 38.89 | 5.71 | 147 | 44.44 | 4.23 | 144 |
| Consistency ineq. | 33.33 | 6.23 | 163 | 38.89 | 4.58 | 149 |
| Subtour elimination ineq. ($ S = 2$) | 33.33 | 6.60 | 162 | 38.89 | 5.10 | 147 |
| Subtour elimination ineq. ($ S = 3$) | 33.33 | 6.45 | 163 | 38.89 | 5.08 | 147 |
| Infeasible path ineq. ($ S = 3$) | 44.44 | 4.13 | 138 | 38.89 | 3.95 | 148 |
| Best setting | 44.44 | 2.82 | 136 | 44.44 | 2.78 | 136 |

Table 4: Analysis of setting for exact approaches based on 18 instances.

| simulated annealing | | regret and worst operator | infeasible initial solutions |
|---------------------|--------|---------------------------|------------------------------|
| w_i | c | η | λ |
| 0.05 | 0.9999 | 0.3 | 2 |

Table 5: LNS parameters.

settings. Simulated annealing parameters, w_i and c , are set to the values reported in [21]. The noise parameter η was adjusted during the development of the algorithm. Parameter λ for adjusting infeasible initial solutions is based on comparisons between the objective values of feasible and infeasible initial solutions. Each instance is solved ten times to account for random variations of the LNS.

6.3. Comparison of exact approaches

In this section, results of the different exact approaches are compared. As already discussed in Section 4, the 4-index model is solved in two ways: directly by CPLEX (*4i-cplex*) and by the B&C algorithm (*4i-bc*). The 3-index model is solved only by the B&C algorithm (*3i-bc*). The B&C algorithm is implemented using control callbacks, which automatically disables features of CPLEX that considerably improve the efficiency of the optimization process (e.g., dynamic search). To allow a fair comparison between the B&C algorithm and solving the 4-index model directly, we also report results of solving the 4-index model by CPLEX with empty control callbacks (*4i-cplex-ec*).

All approaches suffer from memory issues or are unable to solve the root node (i.e., LP-relaxation) within 4 hours for instances with more than 40 users (especially with options f1 and f2). Therefore, the comparison of the approaches is based on all instances with up to 40 users, which ensures that for a certain instance characteristic (e.g., heterogeneity) the same amount of instances are present for each level (homogeneous and heterogeneous). This leads to a total of 864 problems (i.e., 288 instances solved for three levels of Z).

Table 6 compares the four different approaches. It shows the percentage of instances solved to optimality, the average gap between the provided lower bound and the best known solution UB_{bk} , the average computation time in minutes, the average number of nodes in the B&B tree, and the average number of user cuts added to the model. The first part of Table 6 involves all 864 problems, while the next three parts show results for specific values of Z , with 288 instances, respectively. Results per

| Instances | Approach | $Opt(\%)$ | $Gap_{avg}(\%)$ | $CPU(min)$ | # Nodes | # Cuts |
|-----------|--------------------|-----------|-----------------|------------|---------|--------|
| All | <i>4i-cplex</i> | 46.53 | 4.78 | 133 | 15465 | 0 |
| | <i>4i-cplex-ec</i> | 41.44 | 6.06 | 124 | 27790 | 0 |
| | <i>4i-bc</i> | 42.82 | 3.87 | 143 | 6768 | 218 |
| | <i>3i-bc</i> | 50.23 | 2.92 | 126 | 1408 | 1478 |
| $Z = 1$ | <i>4i-cplex</i> | 71.53 | 2.42 | 78 | 1540 | 0 |
| | <i>4i-cplex-ec</i> | 68.40 | 3.08 | 83 | 2874 | 0 |
| | <i>4i-bc</i> | 65.28 | 2.54 | 91 | 729 | 354 |
| | <i>3i-bc</i> | 50.35 | 3.36 | 127 | 2559 | 2609 |
| $Z = 2$ | <i>4i-cplex</i> | 34.38 | 6.49 | 161 | 14100 | 0 |
| | <i>4i-cplex-ec</i> | 29.51 | 7.72 | 156 | 24104 | 0 |
| | <i>4i-bc</i> | 33.68 | 4.88 | 165 | 7654 | 162 |
| | <i>3i-bc</i> | 40.97 | 3.41 | 145 | 1378 | 1121 |
| $Z = 3$ | <i>4i-cplex</i> | 33.68 | 5.44 | 160 | 30759 | 0 |
| | <i>4i-cplex-ec</i> | 26.39 | 7.37 | 134 | 56391 | 0 |
| | <i>4i-bc</i> | 29.51 | 4.19 | 173 | 11922 | 138 |
| | <i>3i-bc</i> | 59.38 | 1.99 | 107 | 286 | 703 |

Table 6: Comparison of exact approaches. Results involve all instances with up to 40 users.

number of users are available in Table A.1 in Appendix.

The approach *3i-bc* gives the best results on average, both in terms of number of instances solved and in terms of relative gap. Several additional observations can be made: (1) applying the B&C algorithm on the 3-index model gives better results than applying it on the 4-index model; (2) the B&C algorithm outperforms CPLEX when comparing *4i-cplex-ec* and *4i-bc*; (3) if the B&C algorithm was integrated in CPLEX (allowing the use of, e.g., dynamic search), even better results could be expected (based on the considerable reduction in solution quality from *4i-cplex* to *4i-cplex-ec*). Combining results of all approaches and all experiments, 498 out of 864 instances (57.64%) could be solved to optimality. The average gap to the best known solution UB_{bk} is 1.79%.

Looking at results for specific values of Z , it is clear that approaches based on the 4-index model perform better than the B&C algorithm on the 3-index model if only a single driver per user is allowed. This may be a result of the weaker link between the assignment and routing variables in the 3-index model compared to the 4-index one, as discussed in Section 3.2. However, if two or more drivers per user are allowed, the 3-index model provides better results. The instance characteristics (f, t, h, and c) have a negligible influence on the efficiency of the exact approaches, except that solution quality of all models decreases considerably when the number of times a trip is active during the planning horizon increases. For example, 69.44% of the f0 instances and 36.11% of the f2 instances could be solved to optimality by the 3-index model (the relative gaps are 0.73% and 5.93%, respectively). A similar observation can be made from Table A.1 in Appendix. The efficiency of all approaches decreases significantly when the number of users (and hence the number of (active) requests) increases. The relative performance of the different approaches is hardly affected by the number of users, except that the advantage of the 3-index model seems to decrease slightly with an increase in the number of users.

6.4. Comparison of exact approaches and LNS

In Table 7, we evaluate the performance of the LNS by comparing its results to the results of the exact algorithms. The table is divided into two parts. The left part (All instances) includes all instances with

| Instances | All instances | | | | Closed instances | | | |
|------------|---------------|-----------------|------------------|-----------------|------------------|-----------------|------------------|-----------------|
| | # inst. | Gap_{avg} (%) | Gap_{best} (%) | CPU_{LNS} (s) | # inst. | Gap_{avg} (%) | Gap_{best} (%) | CPU_{LNS} (s) |
| All | 864 | 3.68 | 3.21 | 88.91 | 498 | 0.35 | 0.13 | 39.21 |
| h0 | 432 | 4.16 | 3.70 | 91.63 | 266 | 0.39 | 0.16 | 43.72 |
| h1 | 432 | 3.20 | 2.72 | 86.18 | 232 | 0.30 | 0.11 | 33.00 |
| t0 | 432 | 1.84 | 1.41 | 83.36 | 257 | 0.34 | 0.14 | 37.86 |
| t1 | 432 | 5.52 | 5.02 | 94.45 | 241 | 0.36 | 0.13 | 39.65 |
| f0 | 288 | 0.81 | 0.52 | 46.82 | 222 | 0.34 | 0.14 | 31.54 |
| f1 | 288 | 2.09 | 1.63 | 86.80 | 152 | 0.35 | 0.13 | 40.01 |
| f2 | 288 | 8.15 | 7.50 | 133.09 | 124 | 0.37 | 0.13 | 50.01 |
| c0 | 288 | 5.11 | 4.61 | 78.05 | 172 | 0.41 | 0.14 | 36.64 |
| c1 | 288 | 3.35 | 2.93 | 85.20 | 164 | 0.33 | 0.15 | 37.20 |
| c2 | 288 | 2.59 | 2.10 | 103.46 | 162 | 0.30 | 0.12 | 42.48 |
| $Z = 1$ | 288 | 2.69 | 2.05 | 87.05 | 206 | 0.55 | 0.21 | 49.75 |
| $Z = 2$ | 288 | 3.82 | 3.39 | 89.37 | 120 | 0.16 | 0.05 | 21.39 |
| $Z = 3$ | 288 | 4.54 | 4.20 | 90.29 | 172 | 0.24 | 0.10 | 37.61 |
| $ U = 5$ | 108 | 0.01 | 0.00 | 4.74 | 108 | 0.01 | 0.00 | 4.74 |
| $ U = 10$ | 108 | 0.09 | 0.01 | 16.83 | 108 | 0.09 | 0.01 | 16.83 |
| $ U = 15$ | 108 | 0.38 | 0.19 | 44.36 | 95 | 0.27 | 0.12 | 41.23 |
| $ U = 20$ | 108 | 0.82 | 0.49 | 66.89 | 74 | 0.44 | 0.15 | 54.32 |
| $ U = 25$ | 108 | 1.57 | 1.05 | 98.40 | 51 | 0.65 | 0.22 | 72.44 |
| $ U = 30$ | 108 | 2.79 | 2.10 | 127.70 | 36 | 1.05 | 0.48 | 81.20 |
| $ U = 35$ | 108 | 8.34 | 7.49 | 161.81 | 20 | 1.05 | 0.50 | 88.35 |
| $ U = 40$ | 108 | 15.46 | 14.38 | 190.50 | 6 | 1.95 | 0.92 | 105.65 |

Table 7: Comparison between LNS and exact approaches. Average LNS and best-out-of-ten results are compared to optimal objective values and best known lower bounds. *All instances* include all instances with up to 40 clients. *Closed instances* include only instances for which the optimal solution is available.

up to 40 users. The LNS results are compared to the best available lower bounds. In the right part of the table (Closed instances), we compare the LNS only to instances where a guaranteed optimal solution is available. The rows show the average results over the instances with the indicated characteristics. Column “# inst.” indicates the number of instances involved in the analysis. The average gap between the best lower bound and the average objective value of the LNS over 10 runs is denoted by Gap_{avg} ; Gap_{best} is the average gap between the lower bound and the best result over the 10 runs.³ CPU_{LNS} is the average computation time per run in seconds. Results per instance are available online.

The performance of the LNS is consistently satisfying. The lower bounds are on average 3.68% better than the average LNS result when all instances are taken into account. The average gap of the best LNS result is 3.21%. On the closed instances, the average gap is 0.35% for the average LNS result and 0.13% for the best LNS result. The solution quality of the LNS is stable across different problem instances. On the closed instances, the LNS generates solutions that are between 0.30% and 0.41% more costly, on average, depending on the specific instance feature. The large gaps for the f2 instances on the left of the table are caused by weak lower bounds rather than poor LNS results.

Looking at different levels of consistency, the average gaps are between 2.69% and 4.54% for the average LNS result depending on the value of Z . On the closed instances, the average objective values

³For each instance, the gap is calculated as $(UB_{LNS} - LB)/LB$, where UB_{LNS} either refers to the best solution or the average solution among ten LNS runs.

are between 0.16% and 0.55% higher than the optimal results. The results on the closed instances might indicate that the LNS performs best when two drivers per user are allowed. These results, however, are biased due to large differences in the number of closed instances for different Z values (i.e., (206, 120, 172) for $Z = (1, 2, 3)$).

Results in the lower part of the table indicate that the LNS scales well. Gaps on the right of the table show that the solution quality of the LNS deteriorates slightly when the number of users increases. On average, however, the LNS generates solutions within 2% of the optimal solution for instances with up to 40 users within about 3 minutes. As before, the large gaps on the left of the table for instances with a large number of users result from weak lower bounds.

The average computation time of the LNS is 88.91 seconds. The large impact of the frequency of requests (f) on the computation time is explained by the increasing number of active requests per user from f0 to f2. The influence of the geographical distribution of the pickup and dropoff locations is unexpected. If locations are distributed randomly, the average computation time is 78.05 seconds. The average computation time increases to 85.20 seconds in the c1 instances and to 103.46 seconds in the c2 instances. The advantage of the LNS over the exact approaches in terms of computation time is obvious. For example, the average time to find the optimal solutions for the 434 instances solved by the B&C algorithm on the 3-index model is 831 seconds, while the LNS generates close to optimum solutions for these instances in less than 30 seconds, on average. Computation time of the LNS is barely affected by the number of different drivers per user. The average computation time is 87.05 seconds when $Z = 1$ and 90.29 seconds when $Z = 3$. Clearly, computation times increase when the number of users increases, as is shown in the lower part of Table 7. Figure 3 shows the average computation times of the LNS on all instances per number of users and level of consistency. Results on the instances with more than 40 users are reported as well. Computation times seem to increase quasi linearly with the number of users, allowing instances of up to 100 users (627 requests on average) to be solved within about 10 minutes on average.

6.5. Cost of driver consistency

In this section, we examine the increase in routing cost when restricting the number of different drivers per user. All 432 instances are solved with the LNS algorithm for this purpose. Table 8 shows the results of comparing the cost of different levels of driver consistency. The first column denotes the instance type. The next three columns show the percentage increase in routing cost if we allow only one, two, and three drivers per user instead of ignoring consistency, respectively. The last two columns show the cost saving that is achieved by transporting users with two rather than one driver and three rather than two drivers, respectively.

The results show that restricting the number of different drivers per user to one is significantly more expensive than allowing at least two drivers. Depending on the instance type, the travel cost increases between 8.52% and 11.96% if $Z = 1$, but only between 0.91% and 1.67% if $Z = 2$. The average increase

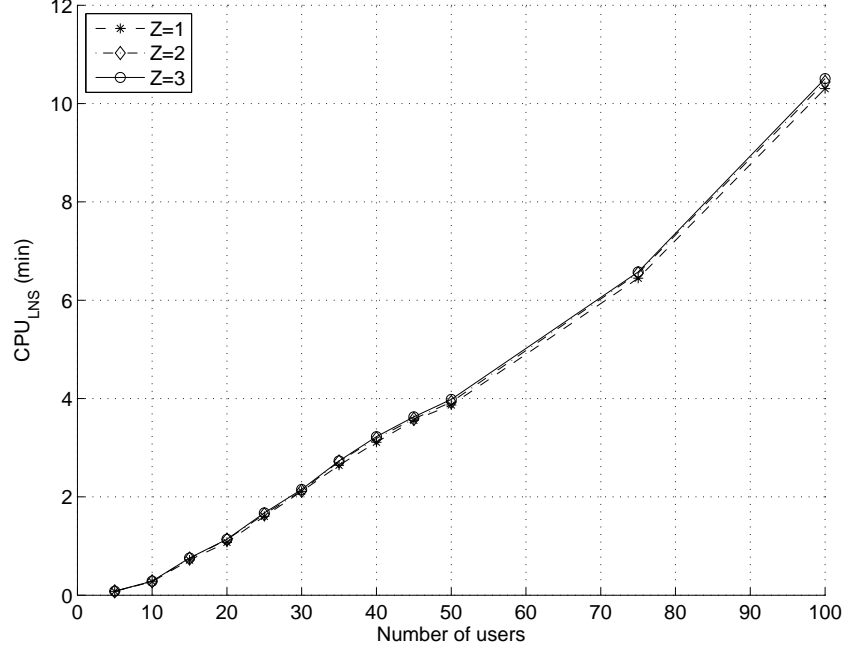


Figure 3: Average computations times of the LNS (in minutes) for different number of users and levels of driver consistency.

in cost that is caused by allowing a single driver per user is 10.70%. The difference between setting Z to two or three is modest: transporting users with three (instead of two) drivers reduces the increase in routing cost by 0.81%, on average. These results are consistent with the literature. For example, Feillet
 585 et al. [28] and Kovacs et al. [24] report average cost savings of 7.5% and 6.5% when driver consistency is relaxed, respectively. Spliet and Dekker [46] demonstrate that visiting each customer by the same driver increases the routing cost by up to 20.9% (12.7%, on average). However, guaranteeing driver consistency to only 75% of the costumers increases cost by 2.9%, on average.

The instance characteristics have a minor effect on the cost of driver consistency. Only the frequency
 590 of trips per user seems to have a significant impact on the routing cost when $Z = 1$. The routing plan becomes 8.52% more costly, on average, if users require a few trips during the planning period (instances f0). However, the increase in cost grows to 11.96% as the frequency of requesting a trip becomes higher (instances f2). Kovacs et al. [25] note that service providers benefit more from a flexible driver-to-user assignment if there is more variation in the demand pattern, i.e., the probability of visiting a customer
 595 on a given day is low. The daily routing plans have to be modified less if there is less variation in the daily demand. For example, the same routing plan could be applied repeatedly if each customer required a visit on each day. Yet, comparing the results of capacitated vehicle routing problems such as the ConVRP to the results of the DC-DARP seems to be unreasonable because of the different notion of customer demand. Customers are visited at home in the ConVRP, while users are transported back
 600 and forth between different locations in the DC-DARP. Additionally, customers are visited at most once per day in the ConVRP, but are transported at least twice in the DC-DARP.

Results per number of users (lower part of Table 8) indicate that the relative cost increase due to

| Instances | Z=1 | Z=2 | Z=3 | Diff. $Z = (1, 2)$ | Diff. $Z = (2, 3)$ |
|-------------|-------|------|------|--------------------|--------------------|
| All | 10.70 | 1.28 | 0.47 | 9.43 | 0.81 |
| h0 | 10.78 | 1.29 | 0.47 | 9.49 | 0.82 |
| h1 | 10.63 | 1.27 | 0.47 | 9.37 | 0.79 |
| t0 | 9.61 | 1.01 | 0.42 | 8.61 | 0.58 |
| t1 | 11.80 | 1.55 | 0.52 | 10.25 | 1.03 |
| f0 | 8.52 | 0.91 | 0.32 | 7.61 | 0.59 |
| f1 | 11.64 | 1.67 | 0.65 | 9.97 | 1.01 |
| f2 | 11.96 | 1.26 | 0.45 | 10.70 | 0.81 |
| c0 | 11.14 | 1.40 | 0.58 | 9.74 | 0.82 |
| c1 | 9.98 | 1.08 | 0.39 | 8.90 | 0.69 |
| c2 | 10.99 | 1.36 | 0.44 | 9.64 | 0.92 |
| $ U = 5$ | 1.59 | 0.00 | 0.00 | 1.59 | 0.00 |
| $ U = 10$ | 4.39 | 0.09 | 0.00 | 4.30 | 0.08 |
| $ U = 15$ | 6.17 | 0.27 | 0.04 | 5.90 | 0.23 |
| $ U = 20$ | 7.68 | 0.60 | 0.10 | 7.09 | 0.49 |
| $ U = 25$ | 8.29 | 0.72 | 0.18 | 7.57 | 0.54 |
| $ U = 30$ | 9.49 | 1.05 | 0.32 | 8.44 | 0.73 |
| $ U = 35$ | 11.59 | 1.29 | 0.48 | 10.31 | 0.81 |
| $ U = 40$ | 12.69 | 1.60 | 0.56 | 11.09 | 1.04 |
| $ U = 45$ | 13.11 | 1.83 | 0.76 | 11.28 | 1.07 |
| $ U = 50$ | 13.89 | 1.76 | 0.69 | 12.12 | 1.07 |
| $ U = 75$ | 18.68 | 2.88 | 1.25 | 15.80 | 1.63 |
| $ U = 100$ | 20.88 | 3.26 | 1.29 | 17.62 | 1.97 |

Table 8: Average increase in routing cost in percent when users are transported with a limited number of different drivers.

imposing driver consistency is larger for instances with more users, i.e., the larger the instance the larger the relative effect of allowing only a limited number of drivers per user. This holds for each level of consistency. For example, the increase in cost for perfect driver consistency ($Z = 1$) is 1.59% on average in the case of 5 users; it is 20.88% on average in the case of 100 users. The results might be biased due to the deteriorating performance of the LNS with increasing number of users. Still, they suggest that especially large dial-ride service providers should carefully consider the level of consistency to offer.

Figure 4 illustrates the results in form of a box-and-whisker diagram. The increase in routing cost of setting Z to one, two, and three instead of a large number is illustrated for three sets of instances. For each set and each Z value, the boxes show the quartiles, the whiskers give the minimum and maximum values, and the asterisks mark the means of the increase in cost compared to a low-cost solution in which consistency is ignored. The left of the figure shows the results of a data set in which each instance characteristic is set to the level that caused the smallest increase in routing cost in Table 8. All instances are considered in the plots in the middle of the figure. At the right, we report the results of another subset of instances where each instance characteristic is set to the level that resulted the largest increase in Table 8.

The increase in cost of assigning one driver per user varies from 0% to 27.98%, depending on the instance at hand; with two drivers, the increase ranges from -1.23% to 5.80% and with three drivers from -0.72% to 2.85%. Negative values result from stochastic variations in the performance of the LNS. The first and third quartiles are 6.11% and 14.85%, respectively, when $Z = 1$, 0.17% and 2.10% when $Z = 2$, and 0% and 0.76% when $Z = 3$. The small difference between the median and mean indicates a

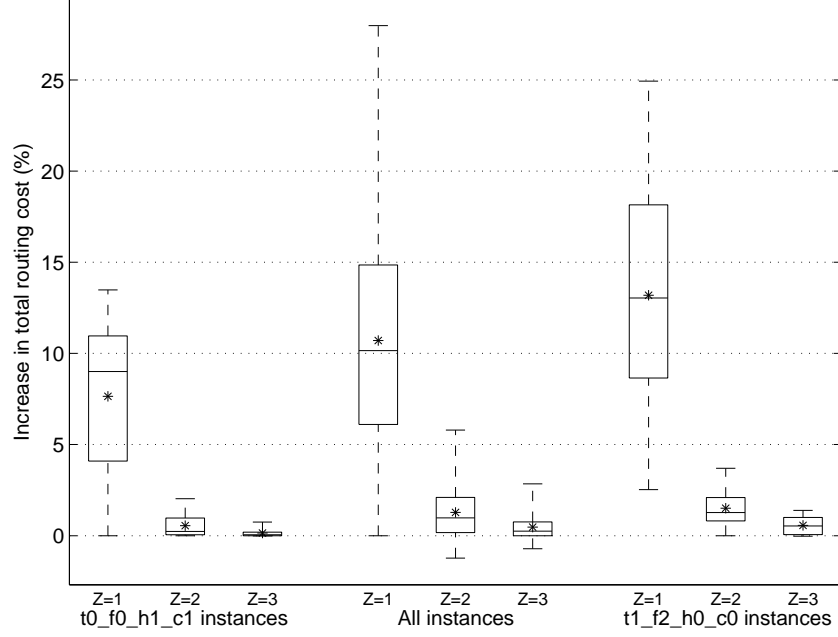


Figure 4: Increase in routing cost for different levels of driver consistency.

symmetric distribution of the values when all instances are taken into account. The median is 10.16% with $Z = 1$, 0.98% with $Z = 2$, and 0.25% with $Z = 3$; the means are 10.70%, 1.28%, and 0.47% respectively.

Even for instances in which driver consistency is relatively cheap (left of Figure 4), we can observe an average deterioration of 7.65% if Z is set to one; the average deterioration is 0.55% with two drivers and 0.15% with three drivers. In most instances, the improvement is on the high end of the scale. In 75% of the instances the improvement is at least 4.09%.

On the right side of the figure, we see that the cost of restricting Z to one driver per user can be up to 13.19% higher, on average, compared to the cost of a loose driver-to-user assignment. Again, the average increase in cost of setting Z to a value larger than one is minor, i.e., 1.51% with $Z = 2$ and 0.57% with $Z = 3$. Given a lower quartile of 8.65% when $Z = 1$, it can be assumed that the increase in cost of providing the highest level of driver consistency would outweigh the benefit of improved user satisfaction in many applications.

We can summarize that there are considerable fluctuations in the cost of driver consistency. Individual instance features seem to have a minor influence on the cost of consistency. However, certain combinations of features can have a significant impact. In some situations, a one-driver-per-user assignment can increase cost by 13.19%, on average. In other situations, an optimal driver consistency with $Z = 1$ can be achieved without having to increase routing cost; i.e., driver consistency comes for free.⁴ Yet, in exceptional cases, a strict driver consistency can raise the routing cost by up to 27.98%. Setting Z to

⁴However, this was observed only in small instances with not more than 10 users and four drivers.

values larger than two provides negligible savings.

Companies planning to create a competitive advantage by transporting users with familiar drivers are advised to carefully assess their operational environment. Selecting a wrong level of consistency might
645 either reduce the quality of service or result in extremely costly solutions.

7. Conclusion

Users of dial-a-ride services are often sensitive to changes in their daily routine. This aspect includes the person who is providing the transportation service, i.e., the driver of the vehicle. Therefore, we have proposed the Driver Consistent Dial-A-Ride Problem (DC-DARP). This problem extends the Dial-
650 A-Ride Problem (DARP) to a multi-period problem in which an additional feature of service quality, referred to as driver consistency, is incorporated. Each user is assigned to a limited number of different drivers over the planning horizon. A diverse set of benchmark instances for the DC-DARP is provided. The instances differ in several aspects, e.g., number of users, frequency of requests, fleet composition, and level of clustering. A five-day planning period is assumed.

Two mathematical formulations and an exact Branch-and-Cut algorithm have been proposed. The
655 first formulation requires routing variables with four indices, but the number of constraints is polynomial in the size of the input. The second formulation requires 3-index routing variables. However, due to the exponential number of constraints, they have to be added in a Branch-and-Cut fashion. Results indicate that the second formulation performs best on average. However, when strict driver consistency is imposed
660 (a single driver per user), the first formulation provides better results on average. In most instances, the Branch-and-Cut algorithm outperforms the state-of-the-art solver CPLEX (version 12.6.1).

A lean heuristic algorithm, based on Large Neighborhood Search, has been introduced to find near-optimal solutions in a short amount of computation time. Only two removal operators and a single repair operator are applied. Still, the algorithm performs well. The lower bounds generated by the exact
665 approaches are on average 3.68% better than the average LNS result. When taking into account only the closed instances, the average gap is 0.35%. The solution quality is stable across different problem instances. The average computation time is less than 90 seconds for instances with up to 40 users, while large-scale instances with up to 100 users (627 requests on average) are solved in about 10 minutes on average.

Results indicate that imposing a strict driver consistency with a single driver per user is significantly
670 more expensive than allowing at least two drivers. Depending on the instance, the routing cost can increase by up to 27.98%, if each user is transported by the same driver each time. The average increase in routing cost of allowing one, two, and three drivers per user compared to a low-cost solution is 10.70%, 1.28%, and 0.47% respectively. Restricting the number of drivers to two seems to be a good compromise
675 between service consistency and routing cost.

Several opportunities for future research exist. We have focused on a short planning horizon, assuming that the routing plan can be executed repeatedly, e.g., each week for several months. Future research may

focus on how driver consistency may be measured and ensured in the long term. Bounding the number of drivers per user might be too rigid. Besides, combined effects of driver consistency and different levels of other service quality aspects (time windows, maximum ride times) may be investigated. Finally, we assumed that driver-to-vehicle assignments are fixed, i.e., a driver is always assigned to the same vehicle. For heterogeneous problems, it might be interesting to study the potential savings of allowing flexible driver-to-vehicle assignments, i.e., a driver may be assigned to different vehicles on different days.

Acknowledgements

The authors would like to thank the anonymous referee for constructive comments and suggestions. This work is supported by the Research Foundation Flanders (FWO) and the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (research project COMEX, Combinatorial Optimization: Metaheuristics & Exact Methods). The computational resources and services used in this work were partly provided by the VSC (Flemish Supercomputer Center), funded by the Hercules Foundation and the Flemish Government – department EWI, and the Vienna Scientific Cluster. This support is gratefully acknowledged.

Appendix A. Additional tables

- [1] Paquette J, Bellavance F, Cordeau JF, Laporte G. Measuring quality of service in dial-a-ride operations: the case of a Canadian city. *Transportation* 2012;39(3):539–64. doi:10.1007/s11116-011-9375-4.
- [2] Paquette J, Cordeau JF, Laporte G. Quality of service in dial-a-ride operations. *Computers & Industrial Engineering* 2009;56(4):1721–34. doi:10.1016/j.cie.2008.07.005.
- [3] Cordeau JF, Laporte G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 2003;37(6):579–94. doi:10.1016/S0191-2615(02)00045-0.
- [4] Cordeau JF, Laporte G. The dial-a-ride problem: models and algorithms. *Annals of Operations Research* 2007;153(1):29–46. doi:10.1007/s10479-007-0170-8.
- [5] Braekers K, Caris A, Janssens GK. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological* 2014;67(1):166–86. doi:10.1016/j.trb.2014.05.007.
- [6] Cordeau JF. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 2006;54(3):573–86. doi:10.1287/opre.1060.0283.

| Instances | Approach | $Opt(\%)$ | $Gap_{avg}(\%)$ | $CPU(min)$ | # Nodes | # Cuts |
|------------|--------------------|-----------|-----------------|------------|---------|--------|
| $ U = 5$ | <i>4i-cplex</i> | 100.00 | 0.00 | 0 | 51 | 0 |
| | <i>4i-cplex-ec</i> | 100.00 | 0.00 | 0 | 923 | 0 |
| | <i>4i-bc</i> | 100.00 | 0.00 | 0 | 16 | 0 |
| | <i>3i-bc</i> | 100.00 | 0.00 | 0 | 0 | 9 |
| $ U = 10$ | <i>4i-cplex</i> | 87.96 | 0.24 | 33 | 32584 | 0 |
| | <i>4i-cplex-ec</i> | 80.56 | 0.55 | 18 | 53596 | 0 |
| | <i>4i-bc</i> | 86.11 | 0.27 | 45 | 17834 | 19 |
| | <i>3i-bc</i> | 100.00 | 0.00 | 1 | 380 | 143 |
| $ U = 15$ | <i>4i-cplex</i> | 70.37 | 0.50 | 76 | 23160 | 0 |
| | <i>4i-cplex-ec</i> | 59.26 | 0.76 | 65 | 72767 | 0 |
| | <i>4i-bc</i> | 63.89 | 0.62 | 93 | 12063 | 59 |
| | <i>3i-bc</i> | 86.11 | 0.11 | 44 | 2092 | 1171 |
| $ U = 20$ | <i>4i-cplex</i> | 46.30 | 1.19 | 130 | 28845 | 0 |
| | <i>4i-cplex-ec</i> | 40.74 | 1.52 | 107 | 29909 | 0 |
| | <i>4i-bc</i> | 44.44 | 1.25 | 143 | 11503 | 193 |
| | <i>3i-bc</i> | 53.70 | 0.89 | 131 | 3309 | 2659 |
| $ U = 25$ | <i>4i-cplex</i> | 30.56 | 1.71 | 173 | 17385 | 0 |
| | <i>4i-cplex-ec</i> | 22.22 | 1.96 | 161 | 48489 | 0 |
| | <i>4i-bc</i> | 25.93 | 1.85 | 185 | 6710 | 277 |
| | <i>3i-bc</i> | 31.48 | 1.75 | 175 | 2708 | 2611 |
| $ U = 30$ | <i>4i-cplex</i> | 21.30 | 2.52 | 199 | 13425 | 0 |
| | <i>4i-cplex-ec</i> | 17.59 | 2.71 | 189 | 8517 | 0 |
| | <i>4i-bc</i> | 13.89 | 2.61 | 211 | 3437 | 309 |
| | <i>3i-bc</i> | 19.44 | 2.81 | 202 | 1828 | 1946 |
| $ U = 35$ | <i>4i-cplex</i> | 11.11 | 12.29 | 219 | 6525 | 0 |
| | <i>4i-cplex-ec</i> | 6.48 | 15.95 | 221 | 6053 | 0 |
| | <i>4i-bc</i> | 5.56 | 8.04 | 231 | 1892 | 443 |
| | <i>3i-bc</i> | 10.19 | 5.99 | 219 | 628 | 1710 |
| $ U = 40$ | <i>4i-cplex</i> | 4.63 | 19.81 | 231 | 1745 | 0 |
| | <i>4i-cplex-ec</i> | 4.63 | 25.01 | 233 | 2067 | 0 |
| | <i>4i-bc</i> | 2.78 | 16.30 | 236 | 692 | 446 |
| | <i>3i-bc</i> | 0.93 | 11.92 | 240 | 318 | 1571 |
| All | <i>4i-cplex</i> | 46.53 | 4.78 | 133 | 15465 | 0 |
| | <i>4i-cplex-ec</i> | 41.44 | 6.06 | 124 | 27790 | 0 |
| | <i>4i-bc</i> | 42.82 | 3.87 | 143 | 6768 | 218 |
| | <i>3i-bc</i> | 50.23 | 2.92 | 126 | 1408 | 1478 |

Table A.1: Comparison of exact approaches per number of users. Results involve all instances with up to 40 users.

- [7] Jain S, Van Hentenryck P. Large neighborhood search for dial-a-ride problems. In: Lee J, editor. Principles and Practice of Constraint Programming - CP 2011. No. 6876 in Lecture Notes in Computer Science; Springer Berlin Heidelberg. ISBN 978-3-642-23785-0; 2011, p. 400–13.
- [8] Paquette J, Cordeau JF, Laporte G, Pascoal MM. Combining multicriteria analysis and tabu search for dial-a-ride problems. Transportation Research Part B: Methodological 2013;52(1):1–16. doi:10.1016/j.trb.2013.02.007.
- [9] Parragh SN, Doerner KF, Hartl RF. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 2010;37(6):1129–38. doi:10.1016/j.cor.2009.10.003.
- [10] Parragh SN. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transportation Research Part C: Emerging Technologies 2011;19(5):912–30. doi:10.1016/j.trc.2010.06.002.
- [11] Parragh SN, Schmid V. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Computers & Operations Research 2013;40(1):490–7. doi:10.1016/j.cor.2012.08.004.

- [12] Ropke S, Cordeau JF, Laporte G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* 2007;49(4):258–72. doi:10.1002/net.20177.
- [13] Liu M, Luo Z, Lim A. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological* 2015;81, Part 1:267–88. doi:10.1016/j.trb.2015.05.009.
- 725 [14] Zhang Z, Liu M, Lim A. A memetic algorithm for the patient transportation problem. *Omega* 2015;54:60–71. doi:10.1016/j.omega.2015.01.011.
- [15] Amirgholy M, Gonzales E. Demand responsive transit systems with time-dependent demand: User equilibrium, system optimum, and management strategy. *Transportation Research Part B: Methodological* 2016;Forthcoming. doi:10.1016/j.trb.2015.11.006.
- 730 [16] Sayarshad HR, Chow JY. A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological* 2015;81, Part 2:539–54. doi:10.1016/j.trb.2015.06.008.
- [17] Cordeau JF, Laporte G. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research* 2003;1(2):89–101. doi:10.1007/s10288-002-0009-8.
- 735 [18] Parragh SN, Doerner KF, Hartl RF. A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft* 2008;58(2):81–117. doi:10.1007/s11301-008-0036-4.
- [19] Doerner K, Salazar-González J. Pickup-and-delivery problems for people transportation. In: Toth P, Vigo D, editors. *Vehicle Routing: Problems, Methods, and Applications*, Second Edition. MOS-SIAM Series on Optimization; Society for Industrial and Applied Mathematics. ISBN 978-1-61197-358-7; 2014, p. 193–212. doi:10.1137/1.9781611973594.ch7.
- 740 [20] Groër C, Golden B, Wasil E. The consistent vehicle routing problem. *Manufacturing & Service Operations Management* 2009;11(4):630–43. doi:10.1287/msom.1080.0243.
- 745 [21] Kovacs AA, Parragh SN, Hartl RF. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks* 2014;63(1):60–81. doi:10.1002/net.21522.
- [22] Tarantilis C, Stavropoulou F, Repoussis P. A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications* 2012;39(4):4233–9. doi:10.1016/j.eswa.2011.09.111.
- 750 [23] Luo Z, Qin H, Che C, Lim A. On service consistency in multi-period vehicle routing. *European Journal of Operational Research* 2015;243(3):731–44. doi:10.1016/j.ejor.2014.12.019.

- [24] Kovacs AA, Golden BL, Parragh SN, Hartl RF. The generalized consistent vehicle routing problem. *Transportation Science* 2015;49(4):796–816. doi:10.1287/trsc.2014.0529.
- [25] Kovacs AA, Parragh SN, Hartl RF. The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research* 2015;247(2):441–58. doi:10.1016/j.ejor.2015.06.030.
- [26] Lian K, Milburn AB, Rardin RL. An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. *IIE Transactions* 2016;Forthcomming. doi:10.1080/0740817X.2016.1167288.
- [27] Subramanyam A, Gounaris CE. A branch-and-cut framework for the consistent traveling salesman problem. *European Journal of Operational Research* 2016;248(2):384–95. doi:10.1016/j.ejor.2015.07.030.
- [28] Feillet D, Garaix T, Lehuédé F, Péton O, Quadri D. A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks* 2014;63(3):211–24. doi:10.1002/net.21538.
- [29] Coelho LC, Cordeau JF, Laporte G. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies* 2012;24(1):270–87. doi:10.1016/j.trc.2012.03.007.
- [30] Coelho LC, Laporte G. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research* 2013;40(2):558–65. doi:10.1016/j.cor.2012.08.012.
- [31] Coelho LC, Laporte G. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research* 2013;51(23-24):7156–69. doi:10.1080/00207543.2012.757668.
- [32] Kovacs AA, Golden BL, Hartl RF, Parragh SN. Vehicle routing problems in which consistency considerations are important: A survey. *Networks* 2014;64(3):192–213. doi:10.1002/net.21565.
- [33] Desrochers M, Laporte G. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters* 1991;10(1):27–36. doi:10.1016/0167-6377(91)90083-2.
- [34] Ruland K, Rodin E. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & Mathematics with Applications* 1997;33(12):1–13. doi:10.1016/S0898-1221(97)00090-4.
- [35] Naddef D, Rinaldi G. Branch-and-cut algorithms for the capacitated VRP. In: Toth P, Vigo D, editors. *The vehicle routing problem*. Philadelphia, PA: SIAM Monographs on Discrete Mathematics and Applications; 2002, p. 53–84.
- [36] Ascheuer N, Fischetti M, Grotschel M. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks* 2000;36(2):69–79. doi:10.1002/1097-0037(200009)36:2<69::AID-NET1>3.0.CO;2-Q.

- 785 [37] Dumas Y, Desrosiers J, Soumis F. The pickup and delivery problem with time windows. *European Journal of Operational Research* 1991;54(1):7–22. doi:10.1016/0377-2217(91)90319-Q.
- [38] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. *Computers & Operations Research* 2007;34(8):2403–35. doi:10.1016/j.cor.2005.09.012.
- 790 [39] Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* 2006;40(4):455–72. doi:10.1287/trsc.1050.0135.
- [40] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget JF, editors. *Principles and Practice of Constraint Programming CP98*; vol. 1520. Springer, Berlin Heidelberg; 1998, p. 417–31.
- 795 [41] Ropke S, Pisinger D. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research* 2006;171(3):750–75. doi:10.1016/j.ejor.2004.09.004.
- [42] Potvin JY, Rousseau JM. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research* 1993;66(3):331–40. doi:10.1016/0377-2217(93)90221-8.
- 800 [43] Cordeau JF, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* 1997;30(2):105–19. doi:10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G.
- [44] Savelsbergh MWP. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing* 1992;4(2):146–54. doi:10.1287/ijoc.4.2.146.
- 805 [45] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. *Science* 1983;220(4598):671–80. doi:10.1126/science.220.4598.671.
- [46] Spliet R, Dekker R. The driver assignment vehicle routing problem. *Networks* 2016;URL: <http://dx.doi.org/10.1002/net.21694>. doi:10.1002/net.21694; available online.