Transportation Research Part B 67 (2014) 166–186

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots

Kris Braekers[a,*], An Caris[a,b], Gerrit K. Janssens[a]

[a] Research Group Logistics, Hasselt University, Campus Diepenbeek, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium
[b] Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussels, Belgium

ARTICLE INFO

Article history:
Received 14 January 2014
Received in revised form 7 May 2014
Accepted 7 May 2014

Keywords:
Dial-a-ride
Demand responsive transportation
Heterogeneous users and vehicles
Branch-and-cut algorithm
Deterministic annealing meta-heuristic
Vehicle routing

ABSTRACT

Dial-a-ride problems are concerned with the design of efficient vehicle routes for transporting individual persons from specific origin to specific destination locations. In real-life this operational planning problem is often complicated by several factors. Users may have special requirements (e.g. to be transported in a wheelchair) while service providers operate a heterogeneous fleet of vehicles from multiple depots in their service area. In this paper, a general dial-a-ride problem in which these three real-life aspects may simultaneously be taken into account is introduced: the Multi-Depot Heterogeneous Dial-A-Ride Problem (MD-H-DARP). Both a three- and two-index formulation are discussed. A branch-and-cut algorithm for the standard dial-a-ride problem is adapted to exactly solve small problem instances of the MD-H-DARP. To be able to solve larger problem instances, a new deterministic annealing meta-heuristic is proposed. Extensive numerical experiments are presented on different sets of benchmark instances for the homogeneous and the heterogeneous single depot dial-a-ride problem. Instances for the MD-H-DARP are introduced as well. The branch-and-cut algorithm provides considerably better results than an existing algorithm which uses a less compact formulation. All seven previously unsolved benchmark instances for the heterogeneous dial-a-ride problem could be solved to optimality within a matter of seconds. While computation times of the exact algorithm increase drastically with problem size, the proposed meta-heuristic algorithm provides near-optimal solutions within limited computation time for all instances. Several best known solutions for unsolved instances are improved and the algorithm clearly outperforms current state-of-the-art heuristics for the homogeneous and heterogeneous dial-a-ride problem, both in terms of solution quality and computation time.

1. Introduction

In dial-a-ride problems, users specify transportation requests from a specific origin location to a specific destination location. Often two related requests are formulated by a user, an outbound request (e.g. from home to a destination) and an inbound request (e.g. return trip to home). These requests are subject to desired pickup or drop off times. To ensure service quality, a maximum user ride time may be imposed as well. The aim is to design a set of efficient routes to perform these transportation requests using a fleet of vehicles with limited capacity.

* Corresponding author. Tel.: +32 11 269120; fax: +32 11 268700.
E-mail addresses: kris.braekers@uhasselt.be (K. Braekers), an.caris@uhasselt.be (A. Caris), gerrit.janssens@uhasselt.be (G.K. Janssens).

http://dx.doi.org/10.1016/j.trb.2014.05.007
0191-2615/© 2014 Elsevier Ltd. All rights reserved.

Dial-a-ride problems arise in the context of demand responsive transportation. A common application is the door-to-door transportation of elderly and disabled people. These people often cannot make use of general public transportation services because these are not adapted to their needs. Other applications of dial-a-ride problems may be found in patient transportation and transportation in rural areas with a lack of general public transportation services. Dial-a-ride services differ from costly taxi services by the fact that users may be grouped together in a vehicle in order to reduce operating costs (Chevrier et al., 2012). Although dial-a-ride services are currently provided in many large cities, these services are expected to become even more widely spread in the future due to the aging population and the trend towards the development of ambulatory health care services (Cordeau and Laporte, 2007; Paquette et al., 2013). Hence, a need for efficient planning tools exists.

In the majority of papers on dial-a-ride problems, a single user type and a homogeneous fleet of vehicles located at a single depot are assumed. In reality, the problem is often more complex due to the presence of users with special requirements and a heterogeneous fleet of vehicles (Mitrović-Minić, 1998; Parragh, 2011; Wong and Bell, 2006). In the context of transporting elderly and disabled people, some users may request to be transported in a wheelchair while others may take a regular vehicle seat. In the context of patient transportation, some users may need to be transported on a stretcher. Moreover, users may often request an accompanying person to join them (Parragh, 2011). To accommodate these varying transportation needs, service providers usually operate a heterogeneous fleet of vehicles. In addition, these vehicles may be stationed at different vehicle depots in their service area (Cordeau and Laporte, 2007; Mitrović-Minić, 1998). Each vehicle should return to its original depot at the end of its route in order to allow the driver to go home by its own means of transport. In some cases, drivers might even be allowed to take the vehicles home at the end of their shift, resulting in the same number of depots as vehicles. Such a situation arises for example when service providers rely on volunteers which use their own passenger car to transport disabled people, a common practice in Belgium (Neven et al., 2014). It is important to take this inherent heterogeneity into account when studying dial-a-ride problems. While heterogeneous users and heterogeneous vehicles have been considered by some authors, the multi-depot aspect has rarely been studied. Furthermore, although these three aspects jointly arise in many real-life applications, to the authors' knowledge only Carnes et al. (2013) study a dial-a-ride problem involving all three aspects (in the context of air-ambulance services).

The contribution of this paper is fourfold. First, a general static dial-a-ride problem is introduced in which heterogeneous users, heterogeneous vehicles and multiple vehicle depots may be taken into account. We will denote this problem the Multi-Depot Heterogeneous Dial-A-Ride Problem (MD-H-DARP). Second, an existing branch-and-cut algorithm for the single depot homogeneous Dial-A-Ride Problem (DARP) is adapted to be applicable to problems with heterogeneous users, heterogeneous vehicles and multiple depots. This branch-and-cut algorithm provides considerably better results for the Heterogeneous DARP (H-DARP) than an existing branch-and-cut algorithm which uses a less compact problem formulation. All previously unsolved benchmark instances for the H-DARP can be solved to optimality within a matter of seconds. Third, a new deterministic annealing meta-heuristic is proposed to solve the DARP, H-DARP and MD-H-DARP. Extensive numerical experiments are presented on benchmark data sets for the single depot DARP and H-DARP. The algorithm clearly outperforms current state-of-the-art meta-heuristics, both in terms of solution quality and computation time. Several best known solutions for unsolved benchmark instances are improved. Finally, benchmark instances are proposed for the multi-depot version of the heterogeneous dial-a-ride problem. These instances are larger and hence more challenging than current benchmark instances for the H-DARP. High quality solutions for these new instances are obtained by both the exact and the meta-heuristic algorithm.

The structure of the paper is as follows. In Section 2, an overview of literature on dial-a-ride problems is presented. Section 3 provides a problem description of the MD-H-DARP, together with a discussion of a three-index and a two-index formulation. The branch-and-cut algorithm and the meta-heuristic algorithms are described in Sections 4 and 5. In Section 6, computational experiments on different sets of benchmark instances are reported. Finally, conclusions are drawn and opportunities for future research are identified in Section 7.

2. Literature review

Detailed overviews of literature on dial-a-ride problems are available in Cordeau and Laporte (2003a, 2007) and Parragh et al. (2008). In the following paragraphs, for the greater part only more recent contributions to the literature are discussed. For a discussion of earlier work we refer to the aforementioned papers. Readers interested in dynamic dial-a-ride problems are referred to Berbeglia et al. (2010) as well.

Due to the application oriented character of dial-a-ride problems, a large variety of objective functions and constraints may be observed in the literature. Together with a lack of benchmark instances in the past, this has complicated the evaluation and comparison of different models and solutions methods. However, Cordeau and Laporte (2003b) provide a rather general problem description, together with a set of benchmark instances (Parragh et al., 2010). Time windows on either the pickup or delivery, maximum ride times and a maximum route duration limit are assumed. The objective is to minimize routing costs. This problem definition has been adopted by several other authors (Cordeau, 2006; Cordeau and Laporte, 2007; Jain and Van Hentenryck, 2011; Paquette et al., 2013; Parragh et al., 2010; Parragh, 2011; Parragh and Schmid, 2013; Ropke et al., 2007). In this paper, the problem definition of the standard DARP of Cordeau and Laporte (2003b) is used as a basis for the MD-H-DARP as well (see Section 3).

Exact solution approaches for dial-a-ride problems are rather scarce. A branch-and-cut algorithm for the standard DARP is proposed by Cordeau (2006). A three-index formulation is used and instances up to four vehicles and 48 requests are solved. Ropke et al. (2007) present another branch-and-cut algorithm for the Pickup and Delivery Problem with Time Windows (PDPTW) and the DARP. Instead of a three-index formulation, two two-index formulations are considered. The first formulation (PDPTW1) is composed of both binary and continuous variables, while the second formulation (PDPTW2) only uses binary variables. Results indicate that the latter outperforms the former formulation, solving instances up to 8 vehicles and 96 requests.

Due to the complexity of the DARP, most research attention is devoted to the development of heuristics and meta-heuristics. Recent insertion heuristics are proposed by Häme (2011) for the single vehicle problem and by Diana and Dessouky (2004) for the multi-vehicle problem. Others have focused on meta-heuristics. A Tabu Search (TS) algorithm for the standard DARP is proposed by Cordeau and Laporte (2003b). A single relocate neighborhood is considered, together with an intra-route operator which is applied after every given number of iterations. Infeasibilities regarding load, time window, ride time and maximum route duration constraints are allowed during the search. A Variable Neighborhood Search (VNS) algorithm for the same problem is proposed by Parragh et al. (2010). The authors use three types of neighborhoods. Infeasible solutions are allowed during the search and solutions worsening the objective value may be accepted with a certain probability. A Large Neighborhood Search (LNS) algorithm is discussed by Jain and Van Hentenryck (2011). The authors use constraint programming to find good reinsertions of randomly selected customers. Recently, Parragh and Schmid (2013) have presented a hybrid column generation and LNS algorithm. Columns with negative reduced costs are generated by a VNS algorithm and different hybridization schemes are studied. Numerical results indicate that this algorithm outperforms the previous ones.

Other authors propose meta-heuristic approaches for problems which slightly differ from that of Cordeau and Laporte (2003b), mostly in terms of the objective function. Dial-a-ride problems with weighted objective functions are studied by Jørgensen et al. (2007), Kirchler and Wolfler Calvo (2013), Wolfler Calvo and Colorni (2007) and Wolfler Calvo and Touati-Moungla (2011). The objectives of minimizing operating cost or maximizing the number of customers serviced are combined with quality of service criteria in the objective function. Minimizing the number of vehicles is considered by Luo and Schonfeld (2007) and Rekiek et al. (2006), while bi-objective (Parragh et al., 2009) and multi-objective (Chevrier et al., 2012; Lehuédé et al., 2014) dial-a-ride problems have been studied as well.

While previous contributions assume homogeneous users and homogeneous vehicles, several authors have taken into account heterogeneous aspects inherent to dial-a-ride problems in practice. A parallel insertion heuristic for a dial-a-ride problem with two types of users and two types of vehicles is presented by Wong and Bell (2006). A heuristic for a large-scale DARP with three types of users, vehicles and drivers is described by Xiang et al. (2006). Driver breaks and maximum driving durations are taken into account. The objective is to minimize a combination of fixed vehicle costs and variable routing, driver, waiting and service related costs. Parragh (2011) provides a standard description of the Heterogeneous DARP (H-DARP) by extending the problem definition of Cordeau and Laporte (2003b) to the case of heterogeneous users and vehicles. A variant of the problem, in which waiting times with users on board are penalized, is described as well. The authors propose both a VNS meta-heuristic and an exact branch-and-cut algorithm using a two-index formulation which is based on the PDPTW1 formulation of Ropke et al. (2007). Four user types and two types of vehicles are considered in the numerical experiments. Instances up to four vehicles and 48 requests are solved. Parragh et al. (2012) introduce a variant of the H-DARP in which driver breaks are taken into account. User ride times are not modeled explicitly but are considered implicitly by imposing time windows at both the pickup and delivery node of each request. A VNS meta-heuristic and a column generation approach are proposed to solve the problem. Finally, both solution methods are integrated into a collaborative scheme. Guerriero et al. (2013) extend the problem definition of Cordeau and Laporte (2003b) by assuming two types of vehicles, regular and extra ones. The latter are more costly than regular vehicles and have a single user capacity. The authors propose a hybrid meta-heuristic, combining Greedy Randomized Adaptive Search (GRASP) with TS. Results indicate that the hybrid algorithm performs better than the TS algorithm on its own. A heterogeneous dial-a-ride problem where vehicles have a configurable capacity is introduced by Qu and Bard (2013). Three user types and four vehicle types are considered. Each vehicle type has multiple possible configurations to divide its capacity over the three user types. Multi-objective DARP with heterogeneous vehicles and users are studied by Atahran et al. (2014) and Paquette et al. (2013). Atahran et al. (2014) propose a multi-objective genetic algorithm for a problem with three objectives, related respectively to transportation costs, service quality and impact on environment. Five types of vehicles are considered, each having a different capacities, costs and emission levels. Paquette et al. (2013) propose a multi-criteria TS algorithm to solve a problem in which routing costs and three objectives related to user convenience are optimized. Two types of users and three types of vehicles are distinguished.

To the authors' knowledge, the only papers explicitly considering multiple vehicle depots in a dial-a-ride context are those of Melachrinoudis et al. (2007) and Carnes et al. (2013). Melachrinoudis et al. (2007) study a practical DARP arising in a health-care organization. Different vehicle capacities, multiple depots and a single user type are assumed. Each vehicle has a predefined origin and destination depot. The objective is to minimize a linear combination of routing costs and user inconvenience. A tabu search heuristic is proposed and the authors analyze the advantage of centralizing vehicle dispatching instead of performing the dispatching separately at each of the depots. Only small numerical experiments on instances with two depots and two to four requests are presented. Carnes et al. (2013) describe a case study for a company providing air-ambulance services using aircrafts located at different depots and having different capacities. In addition, several complicating side constraints are present. Since the problem is relatively small (only ten to twenty requests per day and a maximum of four requests per aircraft), an exact solution method based on a set partitioning formulation is proposed to solve the problem.

Finally, other recent contributions to the literature are made by Diana et al. (2006), Berbeglia et al. (2011) and Häme and Hakula (2013). Diana et al. (2006) propose a model to estimate the number of vehicles required based on the distribution of the demand and desired service quality. Berbeglia et al. (2011) and Häme and Hakula (2013) use respectively constraint programming and a modified version of hyperlink-induced topic search to efficiently check whether a feasible solution to a problem instance exists, i.e. whether all requests can be performed by the vehicles available.

Based on this literature review, it is concluded that a combination of heterogeneous vehicles, heterogeneous users and multiple depots has hardly been studied. However, the MD-H-DARP is highly relevant since these problem characteristics often jointly arise in practice. This paper aims to fill this gap by introducing a general problem description for the MD-H-DARP together with a set of benchmark instances and an exact and a meta-heuristic solution approach.

3. Problem description

The static MD-H-DARP is defined as follows. A number of users request transportation between given origins and destinations. All these requests are known at the start of the planning period. Users may specify a time window on their departure or arrival, but not both. In addition, a route duration limit is imposed on the vehicles and maximum user ride times are considered, indicating the maximum time a user may be in a vehicle (time between end of service at pickup location and start of service at the delivery location). The objective is to find a set of vehicle routes performing all transportation requests while minimizing total distance traveled (Cordeau and Laporte, 2003b).

In the standard DARP, a homogeneous vehicle fleet is assumed. All vehicles have the same capacity and are located at a single depot. Besides, only a single type of users is assumed. The MD-H-DARP constitutes an extension of this problem setting in order to accommodate more realistic assumptions such as heterogeneous users, heterogeneous vehicles and multiple depots. To take into account a heterogeneous set of users, different types of resources are defined for each vehicle (e.g. regular seats and wheelchair places). Each user may require one or more places of each resource type. Since service providers often have different types of vehicles, the maximum capacity level for each resource type might differ between vehicles. Finally, multiple vehicle depots are assumed. Whereas the routing problem is a day-to-day operational decision, the assignment of vehicles to depots may be categorized as a tactical decision on a medium term planning horizon. Therefore, each vehicle is assigned to a specific depot, i.e. it should start and end its route at this depot.

Due to the presence of ride time constraints, in combination with time windows and a route duration limit, the scheduling subproblem and hence checking the feasibility of a solution is more complex than in other routing problems (Parragh and Schmid, 2013). A method to test the feasibility of a schedule for a DARP with a maximum wait time constraint is proposed by Hunsaker and Savelsbergh (2002). An eight step evaluation scheme to determine the feasibility of a given path for the DARP considered in this paper is proposed by Cordeau and Laporte (2003b). The evaluation scheme uses the concept of forward time slack originally proposed by Savelsbergh (1992). In this paper, the adapted version of the evaluation scheme as proposed by Parragh et al. (2010) is applied.

Two formulations of the MD-H-DARP are discussed in the following sections.

3.1. Three-index formulation

A three-index formulation for the standard DARP is provided by Cordeau (2006). This formulation can easily be extended to the MD-H-DARP as follows. First, multiple types of capacity resources and requirements should be considered by adding an index r to all load-related parameters and variables. All constraints in which these parameters and variables appear, should hold (and thus be copied) for every resource type r. Second, to accommodate multiple vehicle depots, for each vehicle k, cost and travel time parameters for arcs to and from the depot should depend on the depot to which the vehicle is assigned.

3.2. Two-index formulation

Branch-and-cut approaches on two-index formulations have been proven to outperform branch-and-cut approaches on three-index formulations for the standard DARP (Ropke et al., 2007) and the H-DARP (Parragh, 2011). Hence, a two-index formulation is presented in this paper as well.

As discussed in Section 2, Parragh (2011) proposes a two-index formulation for the H-DARP which is based on the PDPTW1 formulation of Ropke et al. (2007). To accommodate a heterogeneous fleet of vehicles, artificial origin and destination depots are introduced for each vehicle. This formulation may be adapted to the multi-depot case by adapting the cost and travel time parameters to and from these artificial depots according to the depot to which the vehicle is assigned.

However, in this paper a two-index formulation based on the PDPTW2 formulation of Ropke et al. (2007) is proposed for the H-DARP and MD-H-DARP. This formulation has fewer variables and results in better solutions as is shown in Section 6.

To adapt the PDPTW2 formulation to the MD-H-DARP, for each vehicle a dummy pickup and a dummy delivery vertex are introduced. The first vertex of a route should be a dummy pickup vertex, while the last vertex should be the corresponding dummy delivery vertex. These dummy vertices allow to restrict the available capacities on each route to its appropriate value and to take into account the fact that the distance traveled to the first user pickup and from the last user delivery depends on the depot of the vehicle.

Let $n = n^u + n^v$ with n^u and n^v respectively the number of user requests and the number of vehicles. Furthermore, let $Z = \{1, \ldots, z\}$ denote the set of resource types and $K = \{1, \ldots, n^v\}$ the set of heterogeneous vehicles where a vehicle $k \in K$ has a capacity Q_k^r for resource type $r \in Z$. The maximum capacity for resource type $r \in Z$ over all vehicles is denoted by $Q^r = \max_{k \in K} Q_k^r$. Finally, T_{period} and T_{route} denote respectively the length of the planning period and the route duration limit.

The MD-H-DARP is defined on a complete graph $G = (V, A)$. The vertex set $V = \{0, 2n+1\} \cup P^d \cup P^u \cup D^d \cup D^u$ consists of vertices 0 and $2n+1$ representing an artificial start and end depot, a set of dummy pickup vertices $P^d = \{1, \ldots, n^v\}$, a set of user pickup vertices $P^u = \{n^v + 1, \ldots, n\}$, a set of dummy delivery vertices $D^d = \{n+1, \ldots, n+n^v\}$ and a set of user delivery vertices $D^u = \{n + n^v + 1, \ldots, 2n\}$. To simplify notation, let $P = P^d \cup P^u$ denote the set of pickup vertices and $D = D^d \cup D^u$ the set of delivery vertices. Each pickup vertex $i \in P$ has a corresponding delivery vertex $n+i \in D$. With each vertex $i \in V$ is associated a capacity requirement q_i^r for each resource type r, a non-negative service duration s_i, a maximum ride time L_i and a time window $[e_i, l_i]$ in which service should start.

For the artificial depots $(0, 2n+1)$ and the dummy vertices $(i \in P^d \cup D^d)$ parameter values are as indicated in Table 1. Capacity requirements of a dummy request are set such that the capacity remaining after performing the dummy pickup is equal to the actual capacity of the corresponding vehicle. For all user pickup vertices $(i \in P^u)$ the following holds: $q_i^r \geqslant 0, q_i^r = -q_{n+i}^r, s_i = s_{n+i}, L_i = L_{n+i}$ and either $[e_i, l_i] = [0, T_{period}]$ or $[e_{n+i}, l_{n+i}] = [0, T_{period}]$. The latter indicates that a user can only specify a time window either for the pickup vertex i or for the delivery vertex $n+i$.

Finally, with each arc $(i,j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij} for which the triangle inequality holds. Binary decision variables x_{ij} indicate whether a vehicle travels from vertex i to j. For $i \in P^d, x_{i,n+i} = 1$ indicates that the corresponding vehicle is not used.

Starting from a complete graph, several arcs (i,j) are eliminated from arc set A to ensure that each route starts with a dummy pickup vertex and ends with a dummy delivery vertex:

$$(i, 0), (2n+1, i) \quad \forall i \in V \tag{1}$$

$$(0, j) \quad \forall j \in V \setminus P^d \tag{2}$$

$$(i, 2n+1) \quad \forall i \in V \setminus D^d \tag{3}$$

$$(i, j) \quad \forall i \in P^d, \quad j \in D, \quad j \neq n+i \tag{4}$$

$$(i, j) \quad \forall j \in D^d, \quad i \in P, \quad j \neq n+i \tag{5}$$

The first lines (1)–(3) make sure that the artificial start and end depots $(0, 2n+1)$ are only connected with respectively dummy pickup vertices and dummy delivery vertices. Besides, from a dummy pickup vertex only user pickup vertices and the corresponding dummy delivery vertex may be reached (4), while a dummy delivery vertex may only be reached from user delivery vertices and the corresponding dummy pickup vertex (5).

Before solving the MD-H-DARP, either exactly or heuristically, further arc elimination may be performed in a preprocessing step to reduce the size of the problem. In this paper, all time window tightening and arc elimination procedures as proposed by Dumas et al. (1991) and Cordeau (2006) are applied.

Define $\mathcal{S}$ as the set of all vertex subsets $S \subseteq V$ such that $0 \in S, 2n+1 \notin S$ and there is at least one vertex $i \in P$ for which $i \notin S$ and $n+i \in S$. For any subset $S \subseteq P \cup D$, define $q^r(S) = \sum_{i \in S} q_i^r$. A lower bound on the number of times vehicles must enter and leave S in order to visit all vertices in the set is then provided by $\max\{1, B(S)\}$ with $B(S) = \max_{r \in Z}\{\lceil |q^r(S)|/Q^r \rceil\}$ (Ropke et al., 2007). Finally, denote by $\mathcal{R}$ the set of infeasible paths with respect to time windows, ride times and route duration limits, and let $A(R)$ be the arc set of $R \in \mathcal{R}$.

With these definitions, the MD-H-DARP may be formulated as follows (based on Ropke et al. (2007)):

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{6}$$

Table 1
Parameter values for vertices $i \in \{0, 2n+1\} \cup P^d \cup D^d$.

	0	$2n+1$	P^d	D^d
q_i^r	0	0	$q_i^r = Q^r - Q_k^r$ with $k = i$	$q_i^r = Q_k^r - Q^r$ with $k = i - n$
s_i	0	0	0	0
L_i	T_{period}	T_{period}	T_{route}	T_{route}
$[e_i, l_i]$	$[0, T_{period}]$	$[0, T_{period}]$	$[0, T_{period}]$	$[0, T_{period}]$

Subject to

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0, 2n+1\} \tag{7}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0, 2n+1\} \tag{8}$$

$$\sum_{i,j\in S} x_{ij} \leqslant |S| - 2 \qquad \forall S \in \mathcal{S} \tag{9}$$

$$\sum_{i,j\in S} x_{ij} \leqslant |S| - \max\{1, B(S)\} \quad \forall S \subseteq V \setminus \{0, 2n+1\}, \quad |S| \geqslant 2 \tag{10}$$

$$\sum_{(i,j)\in A(R)} x_{ij} \leqslant |A(R)| - 1 \qquad \forall R \in \mathcal{R} \tag{11}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{12}$$

The objective is to minimize total routing costs (6). Constraints (7) and (8) ensure that each vertex is visited exactly once. Precedence constraints are imposed by inequalities (9). These constraints ensure that each pickup vertex $i \in P$ is visited before its corresponding delivery vertex $n+i \in D$ and that i and $n+i$ are visited by the same vehicle. Inequalities (10) and (11) are respectively rounded capacity constraints (Naddef and Rinaldi, 2002) and infeasible path elimination constraints (Ascheuer et al., 2000). The latter constraints may be strengthened in several ways as indicated by Ropke et al. (2007). Finally, constraints (12) define the binary decision variables. Note that, as a result of arc elimination, it is not necessary to impose a constraint on the number of vehicles used. The only arcs leaving the artificial start depot are those to the dummy pickup vertices which all represent an available vehicle.

4. Branch-and-cut algorithm

To exactly solve the MD-H-DARP, a branch-and-cut algorithm is applied on the two-index formulation presented in the previous section. This algorithm is mainly based on the branch-and-cut algorithm of Ropke et al. (2007) for the standard DARP, although some small adaptations are made and elements of Cordeau (2006) are used as well.

Branch-and-cut algorithms combine a branch-and-bound search procedure with the concept of cutting planes. For a detailed description of branch-and-cut approaches for dial-a-ride problems, the reader is referred to Cordeau (2006); Parragh (2011) and Ropke et al. (2007). In general, a branch-and-cut approach can be described as follows. First, a reduced version of the problem is obtained by removing all constraint families of exponential size, being constraints (9)–(11) in our problem. Second, the LP-relaxation of this reduced problem is solved. The obtained solution represents a lower bound on the original problem. Next, the lower bound is improved by adding cuts to the model and solving the LP-version of the updated model. These cuts represent inequalities which are currently violated but which should be valid for each feasible solution. In addition to inequalities violating the eliminated constraints, other types of violated inequalities might be added. This process continues until no more violated inequalities are found. In case the solution to the current model is integer, the optimal solution to the original problem has been found. In case a fractional solution is obtained, two new problems are created by branching on a fractional variable according to the branch-and-bound principle and the search continues. In this paper, a best-bound strategy is applied and the choice of the branching node is left to CPLEX. Before solving the problem, an upper bound is found by the deterministic annealing meta-heuristic described in Section 5.

Similar as in Ropke et al. (2007), seven types of inequalities are used: precedence, capacity, strengthened infeasible path, generalized order, subtour elimination, fork and reachability constraints. The first three are related to the eliminated constraints (9)–(11) and are required to ensure feasibility. The others may be used to strengthen the formulation. For a detailed discussion of these inequalities, the reader is referred to Ropke et al. (2007). No adaptation of these inequalities is required since the two-index formulation of the MD-H-DARP is equivalent to the PDPTW2 formulation for the standard DARP (only additional dummy vertices have been introduced). In addition, an initial pool of inequalities is generated as proposed by Cordeau (2006). All inequalities in this initial pool are the same as in Cordeau (2006), except those related to precedence constraints. Cordeau (2006) includes precedence constraints between a pickup vertex, a delivery vertex and either the start or end depot. In this paper dummy pickup and dummy delivery vertices are introduced to represent the actual depots, where dummy pickup vertices $i \in P^d$ may only be reached from the artificial start depot 0, and dummy delivery vertices $n+i \in D^d$ only have arcs to the artificial end depot $2n+1$. Therefore, for our problem, it makes more sense the define precedence constraints between a user pickup vertex, a user delivery vertex and either a dummy pickup or dummy delivery vertex instead of the artificial start or end depot. Hence, for each set of vertices $i,j \in P^u, d \in P^d$, the following four types of precedence constraints are included in the initial pool of inequalities:

$$S = \{d, i, n+j\} \Rightarrow x_{di} + x_{i,n+j} + x_{n+j,i} \leqslant 1 \tag{13}$$

$$S = \{i, n+j, n+d\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{n+j,n+d} \leqslant 1 \tag{14}$$

$$S = \{d, i, n+i, n+j\} \Rightarrow x_{di} + x_{i,n+i} + x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} + x_{n+j,n+i} \leqslant 2 \tag{15}$$

$$S = \{i, j, n+j, n+d\} \Rightarrow x_{ij} + x_{ji} + x_{i,n+j} + x_{n+j,i} + x_{j,n+j} + x_{n+j,n+d} \leqslant 2 \tag{16}$$

At each node of the branch-and-bound tree, violated inequalities are found by applying separation procedures. In addition, all inequalities in the initial pool are checked exhaustively at each node, as in Cordeau (2006). For the precedence, capacity, generalized order, fork and reachability constraints the separation procedures are the same as described in Ropke et al. (2007), except that capacity checks have to be performed for each resource type separately. This means that the capacity level of each resource type is maintained and whenever a violation of the capacity of one of the resource types is detected, a violation is found and the corresponding inequality may be added. A similar procedure is used for the strengthened infeasible path constraints as well. Ropke et al. (2007) use a procedure where paths with positive flow in the current solution are constructed, starting from a pickup vertex $i \in P$. Each path is extended as long as the corresponding delivery vertex $(n+i)$ is not reached and a violation is still possible (total flow on the arcs in path R is strictly greater than $|A(R)|-1$ and the path has not reached the end depot). In this paper, paths are extended even when the corresponding delivery vertex is reached (as long as a violation is still possible). In addition, our procedure also checks for capacity violations. Finally, subtour elimination constraints are separated similarly as precedence constraints, in contrast to Cordeau (2006) and Ropke et al. (2007) who use a simple tabu search heuristic. For each request, two maximum flow problems are solved, one between vertices 0 and i and one between vertices $n+i$ and $2n+1$. Arc capacities are equal to the x_{ij} variables for arcs belonging to the minimum arc set (all arcs which may be part of a feasibly path from 0 to i or from $n+i$ to $2n+1$ respectively). Capacities of all other arcs are equal to 0. In case a maximum flow smaller than one is found, a subtour elimination constraint is violated and the corresponding cut is generated.

5. Meta-heuristic algorithm

Hard combinatorial optimization problems, such as the MD-H-DARP, can generally only be solved exactly within an acceptable time frame for relatively small problem instances. In order to solve larger problems, heuristics or meta-heuristics may be applied to find near-optimal solutions. Meta-heuristics provide a more profound search of the objective space than traditional heuristics and are less likely to get stuck in a local optimum since they often allow deteriorating and even infeasible intermediate solutions during the search process. For overviews of meta-heuristics for vehicle routing problems, the reader is referred to Bräysy and Gendreau (2005); Cordeau et al. (2007b,a) and Gendreau et al. (2008).

In this paper, a Deterministic Annealing (DA) meta-heuristic is proposed to solve the MD-H-DARP. Deterministic annealing, also referred to as threshold accepting, is a variant on the well-known simulated annealing meta-heuristic. It was first introduced by Dueck and Scheuer (1990) and can be categorized as a meta-heuristic based on local search (Cordeau et al., 2007b). In each iteration, a neighboring solution $\boldsymbol{x}'$ of current solution $\boldsymbol{x}$ is generated. If the objective value of the new solution is better than that of the current solution, $\boldsymbol{x}'$ is automatically accepted and becomes the new current solution. Otherwise, solution $\boldsymbol{x}'$ is accepted as long as the worsening in the objective value $\Delta = f(\boldsymbol{x}') - f(\boldsymbol{x})$ is smaller than a deterministic threshold value T. This threshold value T is gradually lowered during the search until only solutions improving the objective value are accepted (Caris and Janssens, 2010). Recently, deterministic annealing has been successfully implemented for a number of vehicle routing problems (Braekers et al., 2013, 2014; Bräysy et al., 2003, 2008; Caris and Janssens, 2010; Nikolakopoulos and Sarimveis, 2007; Tarantilis et al., 2004). However, deterministic annealing has not been applied to any type of dial-a-ride problem before.

Deterministic annealing has the advantage that it is relatively easy to understand and implement, and that it mainly relies on only a single parameter (T). This greatly increases its applicability in practice compared to other types of algorithms which are often more complex or require extensive parameter tuning. Besides, the algorithm proposed in this paper does not contain any overly complex components which would prevent it from ever being used in practice. Yet, the proposed algorithm performs extremely well as shown in Section 6.

The following sections contain the details of the meta-heuristic. First, the detailed algorithmic structure and the deterministic annealing scheme to update the threshold T are discussed (Section 5.1). Second, the insertion heuristic to find an initial solution is described (Section 5.2). Finally, the local search operators and details on their implementation are presented in Sections 5.3 and 5.4.

5.1. Algorithmic structure

The structure of the algorithm is presented in Algorithm 1. Let $c(\boldsymbol{x})$ and $k(\boldsymbol{x})$ denote respectively the total distance and the number of vehicles used in solution $\boldsymbol{x}$. An initial solution $\boldsymbol{x_{init}}$ is found by an insertion heuristic (see Section 5.2). This initial solution will be feasible, except that more vehicles than available may be used. Other infeasibilities (e.g. regarding time windows, ride times or route durations) are not considered during any stage of the algorithm. Next, the threshold value T is initialized at its maximum value T_{max} and the best solution $\boldsymbol{x_b}$ and current solution $\boldsymbol{x}$ are set to the initial solution $\boldsymbol{x_{init}}$. The deterministic annealing algorithm is then applied for a predefined number of iterations (n_{iter}). Each iteration consists of two steps. In the first step, several local search operators are applied to alter the current solution. In the second step, the threshold value T is updated and a restart mechanism may be triggered.

Algorithm 1. Algorithmic structure

```
Insertion heuristic to find an initial solution $\boldsymbol{x}_{init}$
Parameter initialization: $T = T_{max}, i_{imp} = 0$ and $\boldsymbol{x} = \boldsymbol{x_b} = \boldsymbol{x_{init}}$
for $i = 1 \rightarrow n_{iter}$ do
  $i_{imp} \leftarrow i_{imp} + 1$
  if $k(\boldsymbol{x}) \leqslant n^v$ then
    $n_{oper} \leftarrow 4$
    $w \leftarrow$ random sample from $\{1, 2, \ldots, 24\}$
  else
    $n_{oper} \leftarrow 5$
    $w \leftarrow$ random sample from $\{1, 2, \ldots, 120\}$
  end if
  for $j = 1 \rightarrow n_{oper}$ do
    Apply local search operator $O_{wj}$ on $\boldsymbol{x}$ to obtain neighboring solution $\boldsymbol{x}'$
    if $(k(\boldsymbol{x}) > n^v$ and $k(\boldsymbol{x}') < k(\boldsymbol{x}))$ or $c(\boldsymbol{x}') < c(\boldsymbol{x}) + T$ then
      $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
      if $(k(\boldsymbol{x_b}) > n^v$ and $k(\boldsymbol{x}) < k(\boldsymbol{x_b}))$ or $c(\boldsymbol{x}) < c(\boldsymbol{x_b})$ then
        $\boldsymbol{x_b} \leftarrow \boldsymbol{x}$
        $i_{imp} \leftarrow 0$
      end if
    end if
  end for
  if $i_{imp} > 0$ then
    $T \leftarrow T - T_{max}/T_{red}$
    if $T < 0$ then
      $r \leftarrow$ random number between 0 and 1
      $T \leftarrow r \times T_{max}$
      if $i_{imp} > n_{imp} \times k(\boldsymbol{x_b})$ then
        $\boldsymbol{x} \leftarrow \boldsymbol{x_b}$
        $i_{imp} \leftarrow 0$
      end if
    end if
  end if
end for
return $\boldsymbol{x_b}$
```

The number of local search operators that is used in the first step depends on the number of vehicles used in the current solution. In case the current number of vehicles is less than or equal to the number available ($k(\boldsymbol{x}) \leqslant n^v$), i.e. a feasible solution to the problem has already been found, four operators aiming at reducing total distance traveled are applied: *relocate*, *exchange*, *2-opt**, *r-4-opt* (see Section 5.3). In case the current solution uses more vehicles than available ($k(\boldsymbol{x}) > n^v$), an additional operator aiming at reducing the number of vehicles is applied as well (*eliminate*). This approach differs from most other state-of-the-art heuristics for dial-a-ride problems which generally do not have a component specifically tailored to reduce the number of vehicles used when this is not a part of the objective function (as is the case in this paper). Instead they allow infeasibilities regarding time window, ride time or route duration constraints to find an initial solution with no more than n^v vehicles and gradually try to remove these infeasibilities (Cordeau and Laporte, 2003b; Parragh et al., 2010).

The local search operators are applied in random order. The number of possible orders W is 4! or 5!, depending on whether the *eliminate* operator is used or not. Let O_{wj} denote the jth operator for possibility w. A random integer $w \in \{1, 2, \ldots, W\}$ is then generated to determine the order of the operators in the current iteration. Each operator returns the best neighbor $\boldsymbol{x}'$ from the current solution $\boldsymbol{x}$ (if a neighbor exists). Solution $\boldsymbol{x}'$ is accepted to become the new current solution when the number of additional vehicles is reduced, or when total distance of the solution is smaller than that of the current solution augmented with the threshold value T. If the neighboring solution $\boldsymbol{x}'$ is accepted, it is checked whether a new global best solution $\boldsymbol{x_b}$ has been found as well.

The second step of an iteration, updating the threshold value T, is only performed when no new global best solution has been found in this iteration. This step is based on the deterministic annealing schemes of Bräysy et al. (2008) and Braekers et al. (2013). The threshold value T is reduced by T_{max}/T_{red}, where T_{red} represents a user defined parameter. Whenever T becomes negative, it is reset to $r \times T_{max}$, with r a random number between zero and one. Finally, the search is restarted from solution $\boldsymbol{x_b}$ when T becomes negative and no improvement has been found for $n_{imp} \times k(\boldsymbol{x_b})$ iterations, where n_{imp} is a user defined parameter.

In case no feasible solution to the problem has been found after the predefined number of iterations, it is concluded that no feasible solution which performs all transportation requests can been found. A solution with n^v vehicles is obtained by moving all requests which are currently not accommodated by one of the available vehicles to a request bank. A second run of the DA algorithm is then applied with the objective of minimizing first the number of requests in the request bank and second total distance. The idea is that the requests in the request bank could be outsourced to a third-party service provider (e.g. a taxi company). An alternative approach to deal with infeasible instances would be to allow the violation of service related constraints (time windows, ride times) at a certain cost.

5.2. Insertion heuristic

To obtain an initial solution to the problem, a basic parallel insertion heuristic is used. For each available vehicle, an empty route is created with the appropriate capacity levels and vehicle depot. All user requests are inserted one by one in their best possible position. When a user request cannot be inserted in one of the existing routes, an additional vehicle is added to the solution. For each resource type, the capacity of this additional vehicle is equal to the maximum capacity level over all available vehicles. In case multiple vehicle depots exist, the coordinates of the depot of the additional vehicle are equal to the average coordinates over all depots.

This heuristic is applied a thousand times to find a good starting solution. In the first iteration, user requests are inserted in increasing order of the start of the time window at the pickup vertex. In the following iterations, user requests are inserted in random order. The best solution found over all iterations is selected as the initial solution for the deterministic annealing meta-heuristic. This best solution is defined as the solution with first the minimum number of additional vehicles and second the minimum total distance.

5.3. Local search operators

Local search operators are used to generate neighbors of the current solution. In this paper, five local search operators are used by the meta-heuristic algorithm: three inter-route operators (*relocate*, *exchange*, *2-opt**), an intra-route operator (*r-4-opt*) and a route elimination operator (*eliminate*). Although more advanced operators, e.g. based on ejection chains, have been proposed for dial-a-ride problems as well (Parragh et al., 2010), it is a deliberate choice to select only elementary operators which are computationally rather inexpensive, allowing a large number of iterations in a limited amount of computation time. The operators are discussed below. Implementation details on how the best possible insertion position of a user request is found and how feasibility checks may be performed efficiently is discussed in Section 5.4.

The *relocate* operator removes a user request from its current route and tries to reinsert it in the best possible position in the route of another vehicle. The operator is applied on each user request of a single randomly selected route. In case the selected route contains only a single request, a solution which uses one vehicle less might be obtained. When the current number of routes is smaller than the number of vehicles available, assigning a request to an idle vehicle, thereby increasing the number of routes, is considered as well.

The *exchange* operator swaps two user requests of two different routes or swaps the vehicles of two routes. Each time the operator is applied, a single route is selected randomly. Next, three types of swaps are considered: swapping any user request of this route with a user request of another route, swapping the vehicle of this route with any different vehicle of another route, and swapping the vehicle of this route with any different idle vehicle. Vehicles may differ from each other in terms of capacity or vehicle depot. In case two user requests are swapped, the pickup (delivery) vertex of the first request may only be inserted at the same position of pickup (delivery) vertex of the second user request in the second route, while the pickup and delivery vertices of the second user request may be inserted at any position in the first route.

The last inter-route operator is the *2-opt** operator proposed by Potvin and Rousseau (1995). This operator removes an arc from two routes and recombines the resulting parts, that is: the first part of route one with the second part of route two and vice versa. This operator is adapted to the problem setting of the MD-H-DARP as follows. Since the pickup and delivery vertices of a user request should be visited by the same vehicle, only arcs for which the vehicle is empty are considered to be removed. Furthermore, since vehicles are heterogeneous, it needs to be checked whether vehicle capacity is sufficient and vehicle depots may be reached in time.

The intra-route operator is a restricted version of the well-known k-opt operator with $k = 4$ (Lin and Kernighan, 1973). In the original k-opt operator, k arcs are removed from a route and the remaining segments are reconnected in all possible ways. In the restricted version used in this paper (*r-4-opt*), only four successive arcs of a route are considered to be removed. This means that the order of three consecutive vertices in a route may change. The *r-4-opt* operator is applied on each set of four successive arcs of a single randomly selected route.

Finally, the *eliminate* operator is used to reduce the number of vehicles used. The operator randomly selects a route and removes all user requests from this route. These user requests are then reinserted in random order in the best possible position in the other routes. If all user requests can be reinserted, a solution with a lower number of vehicles is found. Otherwise, the operator terminates without finding a new solution. Obtaining a solution with the same number of vehicles as the current solution (and potentially lower total distance) is not considered since preliminary results have shown that including this option leads to a considerable increase in computation time while hardly improving solution quality.

5.4. Implementation details

Only feasible solutions are considered during the search of the algorithm and a best improvement strategy is followed. For each operator, all feasible solutions in the neighborhood of the current solution are evaluated and the best among them is selected to become the new current solution, on the condition that this solution is acceptable according to the deterministic annealing acceptance criterion (see Section 5.1). To allow the efficient updating of solutions after local search moves, solutions are maintained in a double-linked list in an array data structure as described by Bräysy et al. (2008).

Checking the feasibility of a route is a complex task due to the presence of ride time constraints and time windows. As discussed in Section 3.2, the adapted version of Parragh et al. (2010) of the eight step evaluation scheme introduced by Cordeau and Laporte (2003b) is used in this paper. However, this evaluation scheme may be time-consuming especially for routes which visit a large number of vertices. To save computation time, the number of calls to this scheme may be reduced by introducing efficient preliminary feasibility checks when evaluating a local search move (Campbell and Savelsberg, 2004; Kindervater and Savelsbergh, 1997). These feasibility checks allow to immediately discard local search moves which violate capacity and time window constraints. The evaluation scheme is only applied for checking ride time and maximum route duration feasibility of routes which are feasible with respect to capacity and time window constraints.

For each vertex $i \in V$, a number of parameters is maintained during the search to allow these preliminary feasibility checks. These parameters include:

- $earliest_i$: the earliest time vertex i can be visited, taking into account travel times and time windows (not ride times).
- $latest_i$: the latest time vertex i can be visited, taking into account travel times and time windows (not ride times).
- cap_i^r: capacity occupation of resource type r after visiting vertex i.
- $maxcap_i^r$: maximum capacity usage of resource type r from vertex i up to the end of the route.

Parameters $earliest_i$ and cap_i^r are calculated by a forward loop through the route under consideration. After a local search move, they need to be updated from the first vertex which has changed in the route up to the end depot. Parameters $latest_i$ and $maxcap_i^r$ are calculated by a backward loop through the route and should be updated from the last vertex which has changed up to the start depot. Let arc (i,j) be part of a solution (i.e. $x_{ij} = 1$), then these parameters may be calculated as follows:

$$earliest_j = \max\{e_j, earliest_i + s_i + t_{ij}\} \tag{17}$$

$$cap_j^r = cap_i^r + q_j^r \quad \forall r \in R \tag{18}$$

$$latest_i = \min\{l_i, latest_j - t_{ij} - s_i\} \tag{19}$$

$$maxcap_i^r = \max\{0, maxcap_j^r + q_i^r\} \quad \forall r \in R \tag{20}$$

Within the insertion heuristic and the *relocate*, *exchange* and *eliminate* operators, user requests should be inserted in their best possible position. Consider the insertion of a request represented by vertices i and $n+i$ in the route of vehicle k. Initially, it is checked whether the request can be accommodated by vehicle k in terms of capacity ($\forall r \in R : q_i^r \leqslant Q_k^r$). If this is the case, all combinations of insertion positions for the pickup and delivery vertices are evaluated by a forward loop through the route. Similar approaches are considered by Paquette et al. (2013) and Xiang et al. (2006). A different approach is used by Cordeau and Laporte (2003b) and Parragh et al. (2010) where the best insertion position of the critical vertex (the one on which a time window is imposed) and its corresponding vertex are determined sequentially. Since more options are considered, our approach may lead to a better solution at the cost of increased computational complexity. To reduce the number of unnecessary computations, the following procedure is applied in our algorithm for each possible insertion position of the pickup vertex i.

First, the feasibility of inserting the pickup vertex i between vertices (pre_i, suc_i) is analyzed. The new earliest arrival time at vertex i is $\widehat{earliest_i} = \max\{e_i, earliest_{pre_i} + s_{pre_i} + t_{pre_i i}\}$. It is checked whether vertex i can be reached in time ($\widehat{earliest_i} \leqslant l_i$) and whether there is no capacity violation at vertex i ($\forall r \in R : cap_{pre_i}^r + q_i^r \leqslant Q_k^r$). Second, the insertion of the delivery vertex $n+i$ at any position after vertex i is considered. In case of inserting $n+i$ directly after i, the effect on total distance is calculated. When this effect is worse than the best insertion position found so far or when it is not acceptable with respect to the deterministic annealing acceptance criterion, the insertion position is discarded. Otherwise, the new arrival times at $n+i$ and suc_i are calculated and it is checked whether both vertices can be reached in time ($\widehat{earliest_{n+i}} \leqslant l_{n+i} \wedge \widehat{earliest_{suc_i}} \leqslant latest_{suc_i}$). Finally, feasibility with respect to ride times and maximum route duration is analyzed using the eight step evaluation scheme. In case of inserting $n+i$ later, it is checked whether all vertices between i and $n+i$ and vertex $n+i$ itself are visited in time and no capacity violations occur. This can be done sequentially, starting with the insertion position the closest to i. The search may be stopped whenever a violation is found, since insertion positions for $n+i$ further down the route will be unfeasible as well. Besides, the search may be stopped whenever $\widehat{earliest_{n+i}} - l_i - s_i > L_i$. Next, it is checked whether the effect on total distance is acceptable and whether the vertex after the insertion position of $n+i$ is visited in time. If this is the case, ride times are checked using the evaluation scheme.

For operator *r-4-opt*, the order of three consecutive vertices between vertices pre_i and suc_i is considered. First, acceptability of a move in terms of distance traveled is analyzed. Second, time window feasibility is maintained by ensuring that the new earliest arrival time at vertex suc_i is at most $latest_{suc_i}$, while capacity feasibility should only be checked for the three involved vertices (capacity levels at pre_i and suc_i will stay the same). Again, the evaluation scheme is only used when a move passes the previous tests.

Finally, consider a local search move of operator *2-opt** on arcs (i, suc_i) and (j, suc_j) traversed by vehicles k_1 and k_2 respectively. The effect on total distance can easily be computed. Furthermore, the move is feasible with respect to capacities when $maxcap^r_{suc_j} \leqslant Q^r_{k_1}$ and $maxcap^r_{suc_i} \leqslant Q^r_{k_2}$ for all resource types r. Efficiently checking time window feasibility is less straightforward since the travel time from the last node of a route to the depot might change due to the multi-depot nature of the problem. Let $\delta(k)$ denote the travel time from the current last node to the end depot for vehicle k. The latest arrival times at suc_j and suc_i are increased with this travel time to ensure that no feasible moves are discarded. The following inequalities are used to discard unfeasible local search moves:

- $earliest_i + s_i + t_{i,suc_j} \leqslant latest_{suc_j} + \delta(k_2)$.
- $earliest_j + s_j + t_{j,suc_i} \leqslant latest_{suc_i} + \delta(k_1)$.

Note that these inequalities represent a necessary but not sufficient condition for time window feasibility, i.e. some unfeasible moves might not violate the inequalities. Hence, for moves which pass this first evaluation procedure, the eight step evaluation scheme is applied to exactly check the feasibility of time windows, ride times and route duration.

6. Computational experiments

In this section, results of extensive numerical experiments are presented. All these experiments are performed on a 2.6 GHz Intel Core laptop with 4 GB RAM. Both algorithms are implemented in C++ and use a single thread. The branch-and-cut algorithm is linked with CPLEX 12.4 using the Concert Technology.

First, the benchmark instances used to test the performance of the proposed exact and meta-heuristic algorithms are described (Section 6.1). In Section 6.2, results of a sensitivity analysis on the parameters of the deterministic annealing meta-heuristic are presented. Sections 6.3, 6.4, 6.5 contain results of the proposed algorithms on the H-DARP, MD-H-DARP and standard DARP. Finally, algorithmic design decisions of the meta-heuristic algorithm are discussed in Appendix A.

6.1. Benchmark instances

Several sets of benchmark instances are available for the standard DARP as defined by Cordeau and Laporte (2003b). A set of twenty randomly generated instances is provided by the same authors (Cordeau and Laporte, 2003b). The number of requests ranges from 24 to 144 of which half of them have a time window on the origin and half of them have a time window on the destination. All vertices are randomly generated in a Euclidean plane of size $[-10, 10]^2$. A single type of user with a capacity requirement of 1 is assumed. For all instances the length of the planning period was set to 1440, the route duration limit to 480, the vehicle capacity to 6 and the maximum user ride time to 90. Time windows range between 15 and 45 for instances R1a–R10a and between 30 and 90 for instances R1b–R10b. Finally, the number of vehicles was set to obtain moderately full routes for instances R1a–R6a and R1b–R6b whereas for instances R7a–R10a and R7b–R10b the number of vehicles is a tight constraint. Best known results on these instances are provided by the hybrid column generation and LNS algorithm of Parragh and Schmid (2013).

Cordeau (2006) proposes two other sets of 12 randomly generated benchmark instances for the standard DARP (sets *a* and *b*). The number of requests ranges from 16 to 48, the number of vehicles from 2 to 4. Again, all vertices are randomly generated in a Euclidean plane of size $[-10, 10]^2$ and half of the requests have a time window on the origin location while the other half have a time window on the destination location. Time windows of 15 min are assumed. For the *a* instances, vehicles have a capacity of 3 while each request has a capacity requirement of 1, a service time of 3 and a maximum ride time of 30. The *b* instances are characterized by a vehicle capacity of 6. Each request has a capacity requirement which is randomly chosen according to a uniform distribution on the set $\{1, \ldots, 6\}$, a service duration equal to the capacity requirement and a maximum ride time of 45. To test their branch-and-cut approaches, Ropke et al. (2007) base themselves on the *a* and *b* instances proposed in Cordeau (2006). For each set, the authors use 9 of the 12 instances and additionally they consider 12 larger instances, with up to 96 requests and 8 vehicles, which are generated in the same way. Optimal solutions for these instances are provided by Ropke et al. (2007), while state-of-the-art heuristic solutions are provided by Parragh and Schmid (2013).

Benchmark instances for the H-DARP are proposed by Parragh (2011). The author proposes three sets of 12 instances by introducing different levels of heterogeneity in the *a* instances provided by Cordeau (2006). The heterogeneous aspects are based on observations made at the Austrian Red Cross (ARC). Up to two types of vehicles per instance are considered. For each vehicle, four types of resources are distinguished: staff seats, patients seats, stretchers and wheelchair places. For each user it is defined whether he/she should be transported seated, on a stretcher or in a wheelchair. Besides, accompanying persons may be present. Accompanying persons may use staff seats, patients seats or may sit on the stretcher while a seated patient may use a patient seat or may sit on the stretcher. Finally, users transported on a stretcher or in a wheelchair may only use the corresponding places. In Table 2 an overview is given of the probabilities used to generate the different types of users for each of the three instance sets. The composition of the vehicle fleet is shown as well. While set *U* considers

Table 2
Benchmark instances of Parragh (2011): probabilities and vehicle fleet.

Instance	Probability for patient to be			Probability for AP	Vehicle fleet
Set	Seated	On a stretcher	In a wheelchair		
U	1.00	0.00	0.00	0.00	homogeneous (T0)
E	0.50	0.25	0.25	0.10	homogeneous (T2)
I	0.83	0.11	0.06	0.50	heterogeneous (T1,T2)

AP: accompanying person; T0: 3 patient seats; T1: 1 staff seat, 6 patient seats, 1 wheelchair place; T2: 2 staff seats, 1 patient seat, 1 stretcher, 1 wheelchair place.

a single user/resource type and a homogeneous vehicle fleet, heterogeneous aspects are introduced in sets *E* and *I*. Data set *E* considers four resource types with a homogeneous fleet of vehicles, whereas data set *I* considers four resource types in combination with a heterogeneous fleet of vehicles.

In this paper, the proposed exact branch-and-cut approach and deterministic annealing meta-heuristic are tested on the three sets of benchmark instances of Parragh (2011) for the H-DARP (*U*,*E*,*I*). To test our solution approaches on the MD-H-DARP, a multi-depot version of these three sets of instances is proposed as well. Instead of a single vehicle depot in the center of the square region $[-10, 10]^2$, four vehicle depots are assumed. These depots are located respectively at the following coordinates: $[-5,-5], [5,5], [-5,5]$ and $[5,-5]$. For each instance, the available vehicles are assigned to the four depots as follows: the first vehicle to the first depot, the second vehicle to the second depot, ..., the fifth vehicle to the first depot, and so on. Since these are relatively small instances (up to 48 requests), we extend each set of multi-depot instances with 12 larger instances similarly as Ropke et al. (2007) did for the original *a* instances of Cordeau (2006). The same probabilities regarding user types are used as in Parragh (2011). These multi-depot instance sets are denoted *U-MD*, *E-MD* and *I-MD* and are available from http://alpha.uhasselt.be/kris.braekers. Finally, results of the proposed deterministic annealing algorithm on the twenty instances of Cordeau and Laporte (2003b) and the 21 *b* instances considered by Ropke et al. (2007) for the standard DARP are reported as well.

6.2. Meta-heuristic performance: sensitivity analysis on parameters

The performance of the deterministic annealing meta-heuristic presented in Section 5 depends on a number of parameters which have to be set in advance. In this section, a sensitivity analysis on these parameters is performed and robust parameter values are identified.

The following parameters have to be set:

- number of iterations (n_{iter}),
- maximum threshold value (T_{max}),
- threshold reduction parameter (T_{red}),
- restart parameter (n_{imp}).

To ensure the robustness of the algorithm and to avoid the need for re-scaling the maximum threshold parameter when applying the algorithm on different types of instances, a relative value for T_{max} is applied. Its value depends on the average distance between any two locations in the network under study ($\bar{c}$), i.e. $T_{max} = \bar{c} \times t_{max}$ with t_{max} the actual parameter to be set.

Based on preliminary computational experiments, the following initial parameter values are selected as a starting point for a sensitivity analysis: $n_{iter} = 250,000$, $t_{max} = 1.5$, $T_{red} = 2,000$ and $n_{imp} = 500$. The analysis is performed on twelve problem instances which are selected such that the number of requests ranges from small to large and different levels of heterogeneity and single and multi-depot instances are present.

First, the number of iterations of the algorithm is analyzed using the initial parameter values. Results of this analysis are shown in Table 3. For ten values of n_{iter}, average gaps with the optimal solution (or best lower bound) over five runs on all instances are presented. Average computation times in seconds are shown as well. Results indicate that solution quality does not improve substantially after 350,000 iterations. Hence, n_{iter} is set to 350,000 for all further experiments.

Second, parameter values for t_{max}, T_{red} and n_{imp} are analyzed. Since results for the value of one parameter might depend on the values of the other two parameters, different combinations of these three parameters are tested. For each parameter 11

Table 3
Number of iterations n_{iter}.

Number of iterations (n_{iter}) (in thousands)	50	100	150	200	250	300	350	400	450	500
Average gap (%)	0.25	0.20	0.18	0.17	0.15	0.14	0.13	0.13	0.12	0.12
Average computation time (s)	11.38	22.74	32.89	44.23	54.13	66.04	76.38	86.92	98.00	109.87

values are considered, resulting in $11^3 = 1331$ parameter combinations. Table 4 shows a summary of the results. The values for each parameter are indicated, with M being a very large number (at least equal to n_{iter}). For each value, the average gap over five runs on all 12 instances over all values of the other parameters are reported. Based on these results, the following parameter values are selected to be used in the remainder of the paper: $t_{max} = 1.2, T_{red} = 300$ and $n_{imp} = 400$. These values yield an average gap of 0.12% on the twelve instances. Besides, the usefulness of adapting the threshold value and applying restarts during the search is demonstrated by the results in Table 4. Average results are considerably worse when the threshold is kept constant ($T_{red} = M$) or no restarts are performed ($n_{imp} = M$). For values other than M, note that parameters T_{red} and n_{imp} seem to have a limited effect on solution quality. This underlines the strength of deterministic annealing as a robust meta-heuristic of which the solution quality mainly depends on a single parameter (t_{max}).

6.3. Results on the H-DARP

In this section, results on the three sets of benchmark instances of Parragh (2011) for the single depot H-DARP are presented. Upgrading conditions as defined by Parragh (2011) are taken into account (e.g. a seated patient may take a patient seat or sit on stretcher).

Table 5 contains the results of the branch-and-cut algorithm described in Section 4 and compares them with the algorithm of Parragh (2011). Column 'z' indicates the proved optimal solution (marked by an asterisk) or the best feasible solution. Column 'bestLB' indicates the best lower bound found. Computation time in seconds, number of nodes in the branch-and-bound tree and number of user cuts added are reported as well. The proposed branch-and-cut algorithm on the PDPTW2 formulation of Ropke et al. (2007) clearly outperforms the algorithm of Parragh (2011) on the PDPTW1 formulation. While the latter algorithm fails to find the optimal solution within 4 h of computation time for 7 instances, our algorithm solves all instances within 13 s. Both the average number of nodes visited in the search tree and the average number of user cuts is much lower for our algorithm. Although a small part of the improvement might be attributed to the use of a more recent version of CPLEX, it is clear that the proposed branch-and-cut algorithm on the compact PDPTW2 formulation provides considerably better results than the branch-and-cut algorithm of Parragh (2011) on the PDPTW1 formulation.

Results of the deterministic annealing algorithm presented in Section 5 are presented in Table 6. A comparison with the VNS algorithm of Parragh (2011) is made. For both algorithms, the best and average solutions over five runs are shown. Relative gaps with the optimal solutions and average computation time in seconds are indicated as well. For 34 out of 36 instances the optimal solution is found in all five runs of the deterministic annealing algorithm. Compared with the VNS algorithm, considerably better results are obtained both in terms of solution quality and computation time.

Finally, note that average computation times of the branch-and-cut algorithm (1s) are considerably smaller than those of the meta-heuristic (18s) for the H-DARP instances. This is no longer true when larger instances are considered, as is shown in the next section.

6.4. Results on the MD-H-DARP

Results of the proposed algorithms on three sets of multi-depot instances (*U-MD*, *E-MD* and *I-MD*) are presented in this section. Table 7 contains results on the smaller instances (16–48 requests, 2–4 vehicles) while Table 8 contains results on the larger instances (40–96 requests, 5–8 vehicles). Results are reported similarly as in the previous section. For instances for which the optimal solution is unknown, gaps with the best lower bound ('bestLB') are reported.

For the smaller instances, the branch-and-cut algorithm finds optimal solutions for all instances within a minute of computation time. The meta-heuristic algorithm finds the optimal solution in all five runs for 34 out of 36 instances. Gaps for the other instances are very small. Based on these results, it may be concluded that extending the problem to a multi-depot context has almost no effect on the performance of both algorithms.

For the larger instances, the branch-and-cut algorithm is no longer able to find optimal solutions for all instances within a four hour time limit (29 out of 36 instances are solved to optimality). However, high-quality solutions are obtained by the meta-heuristic algorithm within a limited time frame (40 s on average). The average gap with the optimal solution or best lower bound is only 0.45%. Furthermore, note that for all instances which are solved to optimality, average gaps over five runs are all within 0.56% of the optimal solution. These results indicate that when instances are too large to be solved exactly

Table 4
Results of sensitivity analysis on t_{max}, T_{red} and n_{imp}.

t_{max}	Values	0.3	0.6	0.9	1.2	1.5	1.8	2.1	2.4	2.7	3.0	M
	Avg. gaps (%)	1.91	0.49	0.29	0.17	0.17	0.18	0.19	0.19	0.20	0.20	0.60
T_{red}	Values	100	200	300	400	500	1000	2000	3000	4000	5000	M
	Avg. gaps (%)	0.42	0.40	0.39	0.39	0.39	0.39	0.40	0.39	0.40	0.40	0.66
n_{imp}	Values	100	200	300	400	500	1000	2000	3000	4000	5000	M
	Avg. gaps (%)	0.41	0.41	0.41	0.41	0.41	0.41	0.42	0.42	0.43	0.42	0.47

Table 5
H-DARP: branch-and-cut results on instances of Parragh (2011).

Instance	Results of Parragh (2011)[a]					Own results				
	z	bestLB	CPU	Nodes	Cuts	z	bestLB	CPU	Nodes	Cuts
U										
a2–16	294.25*	294.25	1	0	44	294.25*	294.25	0	0	66
a2–20	344.83*	344.83	3	0	112	344.83*	344.83	0	0	120
a2–24	431.12*	431.12	9	0	161	431.12*	431.12	0	1	175
a3–18	300.48*	300.48	5	0	249	300.48*	300.48	0	0	107
a3–24	344.83*	344.83	8	0	422	344.83*	344.83	1	0	206
a3–30	494.85*	494.85	10	0	575	494.85*	494.85	0	0	285
a3–36	583.19*	583.19	105	0	1099	583.19*	583.19	1	2	307
a4–16	282.68*	282.68	6	0	430	282.68*	282.68	1	1	218
a4–24	375.02*	375.02	6	0	347	375.02*	375.02	0	0	205
a4–32	485.50*	485.50	31	0	1056	485.50*	485.50	1	3	500
a4–40	557.69*	557.63	8328	9723	6293	557.69*	557.69	2	4	474
a4–48	668.82	664.64	14543	5076	5164	668.82*	668.82	2	4	767
$\overline{U}$		429.92	1921	1233	1329	430.27	430.27	1	2	286
E										
a2–16	331.16*	331.13	284	4908	1461	331.16*	331.16	0	0	44
a2–20	347.03*	347.03	8	0	150	347.03*	347.03	0	0	83
a2–24	450.25*	450.21	891	5743	1298	450.25*	450.25	0	0	134
a3–18	300.63*	300.63	4	0	304	300.63*	300.63	0	0	110
a3–24	344.91*	344.91	10	0	505	344.91*	344.91	0	0	164
a3–30	500.58*	500.53	1609	4898	2600	500.58*	500.58	0	0	313
a3–36	583.19*	583.19	101	0	1133	583.19*	583.19	0	0	174
a4–16	285.99*	285.99	759	7664	1832	285.99*	285.99	0	0	195
a4–24	383.84	380.48	14472	19808	22083	383.84*	383.84	0	0	164
a4–32		488.14	14494	4733	12610	500.24*	500.24	1	0	359
a4–40		558.09	14519	2390	17918	580.42*	580.42	1	0	273
a4–48		665.02	14540	1476	11919	670.52*	670.52	2	3	667
$\overline{E}$		436.28	5140	4302	6151	439.90	439.90	<1	<1	223
I										
a2–16	294.25*	294.25	1	0	25	294.25*	294.25	0	0	0
a2–20	355.74*	355.71	115	1103	428	355.74*	355.74	0	3	225
a2–24	431.12*	431.12	7	0	236	431.12*	431.12	1	0	214
a3–18	302.17*	302.15	31	187	515	302.17*	302.17	1	3	209
a3–24	344.83*	344.83	7	0	436	344.83*	344.83	1	2	246
a3–30	494.85*	494.85	10	0	574	494.85*	494.85	0	0	346
a3–36	618.63	592.69	14500	8742	17455	618.15*	618.15	3	18	659
a4–16	299.05*	299.05	37	35	606	299.05*	299.05	1	4	363
a4–24	375.02*	375.02	5	0	391	375.02*	375.02	1	0	274
a4–32	486.93*	486.93	41	0	1093	486.93*	486.93	2	3	726
a4–40	557.69*	557.63	2600	2093	4692	557.69*	557.69	3	3	584
a4–48	678.59	663.30	14498	1820	9526	670.72*	670.72	13	18	1191
$\overline{I}$		433.13	2654	1165	2998	435.88	435.88	2	5	420
$\overline{UEI}$		411.64	9957	4971	3493	435.35	435.35	1	2	310

[a] Results on 3.2 GHz Pentium D computer with 4 GB RAM using CPLEX 11.0.

by the branch-and-cut algorithm within an acceptable time frame, the deterministic annealing algorithm may be applied to quickly find good, near-optimal solutions.

6.5. Results on the homogeneous/standard DARP

To further illustrate the strength of the proposed meta-heuristic algorithm, it is tested on the twenty instances of Cordeau and Laporte (2003b) and the 21 *b* instances considered by Ropke et al. (2007) for the standard DARP. Results are shown in Tables 9 and 10. A comparison is made with results of the recent hybrid LNS algorithm of Parragh and Schmid (2013) which to the authors' knowledge currently provides the best heuristic results on these instances. Best and average results over five runs are presented, together with gaps with the best known ('BKS') or optimal solution.

Results in Tables 9 and 10 show that the proposed algorithm offers high-quality solutions for the standard DARP as well. Results are on average better than those of the hybrid algorithm of Parragh and Schmid (2013). For both instance sets, average gaps over five runs of our algorithm are respectively 0.81% and 0.01% compared to 1.42% and 0.12% for the hybrid LNS algorithm. Furthermore, computation times are considerably smaller, especially for large instances. Computation times are reduced by more than a factor 10 for instances with more than 100 user requests (R5a/b, R6a/b, R9a/b, R10a/b). Finally,

Table 6
H-DARP: meta-heuristic results on instances of Parragh (2011).

Instance	VNS of Parragh (2011)[a]					DA algorithm				
	Best	Gap	Avg.	Gap	CPU	Best	Gap	Avg.	Gap	CPU
U										
a2–16	294.25	0.00	294.25	0.00	68.2	294.25	0.00	294.25	0.00	8.6
a2–20	344.83	0.00	344.83	0.00	133.8	344.83	0.00	344.83	0.00	20.2
a2–24	431.12	0.00	431.12	0.00	187.8	431.12	0.00	431.12	0.00	17.4
a3–18	300.48	0.00	300.48	0.00	45.4	300.48	0.00	300.48	0.00	9.4
a3–24	344.83	0.00	345.26	0.12	86.8	344.83	0.00	344.83	0.00	16.6
a3–30	494.85	0.00	495.11	0.05	105.6	494.85	0.00	494.85	0.00	18.8
a3–36	583.30	0.02	583.89	0.12	162.6	583.19	0.00	583.19	0.00	28.4
a4–16	282.68	0.00	282.68	0.00	26.0	282.68	0.00	282.68	0.00	9.8
a4–24	375.02	0.00	375.04	0.01	50.8	375.02	0.00	375.02	0.00	13.0
a4–32	486.88	0.28	488.27	0.57	86.0	485.50	0.00	485.50	0.00	25.6
a4–40	561.80	0.74	565.58	1.42	130.6	557.69	0.00	557.69	0.00	26.4
a4–48	673.64	0.72	680.98	1.82	253.8	668.82	0.00	668.82	0.00	35.4
$\overline{U}$	431.14	0.15	432.29	0.35	111.45	430.27	0.00	430.27	0.00	19.13
E										
a2–16	331.16	0.00	331.16	0.00	65.6	331.16	0.00	331.16	0.00	9.4
a2–20	347.03	0.00	347.03	0.00	120.0	347.03	0.00	347.03	0.00	19.6
a2–24	450.25	0.00	450.25	0.00	160.4	450.25	0.00	450.25	0.00	15.8
a3–18	300.63	0.00	300.63	0.00	47.6	300.63	0.00	300.63	0.00	9.6
a3–24	344.91	0.00	345.59	0.20	76.2	344.91	0.00	344.91	0.00	14.6
a3–30	500.58	0.00	501.41	0.17	107.6	500.58	0.00	500.58	0.00	17.0
a3–36	583.19	0.00	583.79	0.10	161.6	583.19	0.00	583.19	0.00	23.6
a4–16	285.99	0.00	285.99	0.00	25.0	285.99	0.00	285.99	0.00	8.2
a4–24	383.84	0.00	384.03	0.05	52.6	383.84	0.00	383.84	0.00	12.2
a4–32	502.52	0.45	504.79	0.91	83.0	500.24	0.00	500.24	0.00	22.8
a4–40	585.64	0.90	588.40	1.38	121.0	580.42	0.00	580.42	0.00	24.2
a4–48	675.37	0.72	681.80	1.68	252.2	670.52	0.00	670.52	0.00	33.6
$\overline{E}$	440.93	0.17	442.07	0.37	106.07	439.90	0.00	439.90	0.00	17.55
I										
a2–16	294.25	0.00	294.25	0.00	68.4	294.25	0.00	294.25	0.00	7.2
a2–20	355.74	0.00	355.74	0.00	141.8	355.74	0.00	355.74	0.00	17.4
a2–24	431.12	0.00	431.12	0.00	211.0	431.12	0.00	431.12	0.00	12.6
a3–18	302.17	0.00	302.17	0.00	47.2	302.17	0.00	302.17	0.00	8.4
a3–24	344.83	0.00	344.99	0.05	83.6	344.83	0.00	344.83	0.00	13.4
a3–30	494.85	0.00	495.13	0.06	106.8	494.85	0.00	494.85	0.00	14.8
a3–36	618.58	0.07	619.64	0.24	170.6	618.15	0.00	618.59	0.07	22.6
a4–16	299.05	0.00	299.05	0.00	27.0	299.05	0.00	299.05	0.00	7.2
a4–24	375.07	0.01	376.19	0.31	51.6	375.02	0.00	375.02	0.00	12.0
a4–32	486.93	0.00	488.64	0.35	88.0	486.93	0.00	487.50	0.12	21.0
a4–40	561.35	0.66	563.34	1.01	132.2	557.69	0.00	557.69	0.00	23.8
a4–48	680.43	1.45	687.44	2.49	262.4	670.72	0.00	670.72	0.00	30.0
$\overline{I}$	437.03	0.18	438.14	0.38	115.88	435.88	0.00	435.96	0.02	15.87
$\overline{UEI}$	436.37	0.17	437.50	0.36	111.13	435.35	0.00	435.38	0.01	17.52

[a] Results on 3.2 GHz Pentium D computer with 4 GB RAM.

during algorithm testing, the best known solution for six out of twenty instances of Cordeau and Laporte (2003b) was improved by respectively 0.12%, 0.39%, 0.12%, 0.95%, 0.58% and 0.95%. Objective values of these solutions are shown in the final column of Table 9 while they are available in detail from http://alpha.uhasselt.be/kris.braekers.

7. Conclusions and future research

Dial-a-ride problems differ from other vehicle routing problems due to the human perspective which should be accounted for. Major applications are the door-to-door transportation of elderly and disabled people, patient transportation and transportation in rural areas with a lack of general public transportation services. Dial-a-ride services are expected to become more widely spread in the future due to the aging population and the trend towards ambulatory health care services.

Most research has focused on dial-a-ride problems with homogeneous users and vehicles. In practice, service providers often operate a heterogeneous fleet of vehicles to transport several types of users (e.g. seated or in wheelchair). Furthermore, vehicles may be situated at different vehicle depots and need to return to their origin depot at the end of the day. In some cases drivers might even be allowed to take the vehicle home after their shift. In this paper, a general dial-a-ride problem combining all three real-life aspects of the problem is introduced (denoted MD-H-DARP). Its characteristics are based on

Table 7
MD-H-DARP: branch-and-cut and meta-heuristic results on small instances.

Instance	Branch-and-cut algorithm					DA algorithm				
	z	bestLB	CPU	Nodes	Cuts	Best	Gap	Avg.	Gap	CPU
U-MD										
a2–16	284.18*	284.18	0	0	74	284.18	0.00	284.18	0.00	5.4
a2–20	343.43*	343.43	0	0	106	343.43	0.00	343.43	0.00	16.0
a2–24	427.17*	427.17	1	4	199	427.17	0.00	427.17	0.00	15.8
a3–18	289.67*	289.67	1	5	200	289.67	0.00	289.67	0.00	8.0
a3–24	348.30*	348.30	1	31	378	348.30	0.00	348.30	0.00	14.6
a3–30	469.16*	469.16	0	0	322	469.16	0.00	469.16	0.00	14.4
a3–36	592.42*	592.42	2	3	355	592.42	0.00	592.42	0.00	24.6
a4–16	262.44*	262.44	1	7	369	262.44	0.00	262.44	0.00	7.2
a4–24	355.72*	355.72	1	2	265	355.72	0.00	355.72	0.00	10.8
a4–32	461.65*	461.65	2	6	539	461.65	0.00	461.65	0.00	17.8
a4–40	540.34*	540.34	3	9	646	540.34	0.00	540.34	0.00	20.0
a4–48	631.75*	631.75	5	5	960	631.75	0.00	632.31	0.09	25.8
$\overline{U-MD}$	417.18	417.18	1	6	368	417.18	0.00	417.23	0.01	15.0
E-MD										
a2–16	327.67*	327.67	0	0	48	327.67	0.00	327.67	0.00	6.4
a2–20	345.59*	345.59	1	0	69	345.59	0.00	345.59	0.00	17.8
a2–24	445.88*	445.88	1	0	139	445.88	0.00	445.88	0.00	15.2
a3–18	289.67*	289.67	1	3	213	289.67	0.00	289.67	0.00	7.8
a3–24	348.61*	348.61	1	15	239	348.61	0.00	348.61	0.00	12.4
a3–30	471.43*	471.43	1	0	303	471.43	0.00	471.43	0.00	12.8
a3–36	593.84*	593.84	2	3	219	593.84	0.00	593.84	0.00	20.8
a4–16	262.44*	262.44	0	0	182	262.44	0.00	262.44	0.00	6.0
a4–24	365.54*	365.54	1	3	302	364.54	0.00	364.54	0.00	10.4
a4–32	476.59*	476.59	2	4	515	476.59	0.00	476.59	0.00	16.2
a4–40	562.86*	562.86	2	4	472	562.86	0.00	562.86	0.00	19.4
a4–48	633.49*	633.49	5	0	674	633.49	0.00	633.49	0.00	25.0
$\overline{E-MD}$	426.88	426.88	1	3	281	426.88	0.00	426.88	0.00	14.2
I-MD										
a2–16	284.18*	284.18	0	0	79	284.18	0.00	284.18	0.00	7.6
a2–20	358.88*	358.88	1	3	266	358.88	0.00	358.88	0.00	19.6
a2–24	439.29*	439.29	1	1	214	439.29	0.00	439.29	0.00	13.8
a3–18	292.41*	292.41	0	3	263	292.41	0.00	292.41	0.00	7.8
a3–24	348.54*	348.54	0	2	298	348.54	0.00	348.54	0.00	13.2
a3–30	486.04*	486.04	1	0	343	486.04	0.00	486.04	0.00	14.8
a3–36	626.96*	626.96	4	31	636	626.96	0.00	627.73	0.12	23.6
a4–16	285.40*	285.40	1	1	404	285.40	0.00	285.40	0.00	6.0
a4–24	357.51*	357.51	1	3	329	357.51	0.00	357.51	0.00	12.0
a4–32	471.54*	471.54	3	4	704	471.54	0.00	471.54	0.00	16.6
a4–40	542.56*	542.56	6	10	866	542.56	0.00	542.56	0.00	21.4
a4–48	637.58*	637.58	49	201	1753	637.58	0.00	637.58	0.00	26.8
$\overline{I-MD}$	427.57	427.57	6	22	513	427.57	0.00	427.64	0.01	15.2
$\overline{UEI-MD}$	423.88	423.88	3	10	387	423.88	0.00	423.92	0.01	14.8

a widely used problem definition for the standard DARP. A three- and two-index formulation of the MD-H-DARP are discussed and a set of benchmark instances is proposed.

Two algorithms are presented to solve the MD-H-DARP: an exact branch-and-cut algorithm and a deterministic annealing meta-heuristic. The branch-and-cut algorithm is based on a similar algorithm for the standard DARP. It is applied on a compact two-index formulation using only binary variables. Results on several sets of benchmark instances indicate that the algorithm finds optimal solutions within a matter of seconds as long as the number of user requests is limited (about 50). For larger problems, computation times increase drastically. The algorithm provides better results than an existing branch-and-cut algorithm for the H-DARP using a two-index formulation with both binary and continuous variables.

The proposed deterministic annealing meta-heuristic is a local-search based algorithm in which solutions which deteriorate the objective value are accepted according to an adaptive, deterministic threshold. Several types of basic neighborhoods are applied, including one which specifically aims at reducing the number of vehicles in the solution. Implementation details regarding feasibility checks and local search moves are presented and a sensitivity analysis on the algorithm parameters is reported. The algorithm has the advantage that it is easy to understand, does not contain any overly complex components, mainly relies on a single parameter, and yet provides high-quality results in limited computation time. This greatly increases its applicability in practice compared to other algorithms which are often more complex or require extensive parameter tuning. Results show that near-optimal solutions are obtained within a couple of minutes for all benchmark

Table 8
MD-H-DARP: branch-and-cut and meta-heuristic results on large instances.

Instance	Branch-and-cut algorithm					DA algorithm				
	z	bestLB	CPU	Nodes	Cuts	Best	Gap	Avg.	Gap	CPU
U-MD										
a5–40	482.19*	482.19	7	46	1007	482.19	0.00	482.19	0.00	19.8
a5–50	664.54*	664.54	29	96	1521	664.54	0.00	665.17	0.09	28.6
a5–60	789.87*	789.87	22	17	1670	789.87	0.00	789.87	0.00	37.8
a6–48	586.08*	586.08	29	41	1828	586.08	0.00	586.08	0.00	25.2
a6–60	776.63*	776.63	147	112	3048	776.63	0.00	776.65	0.00	32.2
a6–72	883.78*	883.78	678	543	5857	883.78	0.00	883.78	0.00	49.2
a7–56	680.08*	680.08	141	78	3328	680.08	0.00	682.14	0.30	28.4
a7–70	854.22*	854.22	2913	1575	8384	855.76	0.18	857.67	0.40	40.4
a7–84	1007.33*	1007.33	12484	6143	18175	1007.33	0.00	1009.92	0.26	51.8
a8–64	713.11*	713.11	1427	840	8966	713.11	0.00	713.11	0.00	35.6
a8–80	890.66	885.45	14401	1289	19981	890.69	0.59	892.79	0.83	47.6
a8–96	1185.45	1170.91	14403	2140	10366	1187.26	1.40	1189.74	1.61	61.0
$\overline{U-MD}$		791.18	3890	1077	7011	793.11	0.18	794.09	0.29	38.1
E-MD										
a5–40	483.84*	483.84	7	21	956	483.84	0.00	483.84	0.00	19.4
a5–50	674.19*	674.19	12	27	1227	674.19	0.00	674.71	0.08	26.8
a5–60	813.96*	813.96	86	76	2411	813.96	0.00	814.00	0.00	34.6
a6–48	599.76*	599.76	128	279	4035	599.76	0.00	599.91	0.03	24.2
a6–60	802.49*	802.49	36	12	1886	802.49	0.00	803.59	0.14	34.4
a6–72	915.03*	915.03	101	67	2496	915.03	0.00	915.53	0.05	50.0
a7–56	703.62*	703.62	124	107	2967	703.62	0.00	703.62	0.00	28.2
a7–70	910.91*	910.91	403	165	5132	911.06	0.02	913.51	0.29	42.0
a7–84	1059.12*	1059.12	3765	1104	15336	1060.21	0.10	1062.97	0.36	51.8
a8–64	731.11*	731.11	110	26	2886	731.11	0.00	733.01	0.26	44.0
a8–80	925.72*	925.72	1704	340	8403	927.89	0.23	930.95	0.56	71.2
a8–96	1220.78	1214.69[a]	14402	669	13196	1222.43	0.58	1225.02	0.79	94.4
$\overline{E-MD}$		819.54	1740	241	5078	820.47	0.08	821.72	0.21	43.4
I-MD										
a5–40	496.36*	496.36	40	69	1554	496.36	0.00	496.36	0.00	18.8
a5–50	669.30*	669.30	96	80	2172	669.30	0.00	669.67	0.06	27.4
a5–60	800.10*	800.10	40	13	1725	800.10	0.00	802.42	0.29	37.6
a6–48	586.08*	586.08	35	25	1905	586.08	0.00	586.08	0.00	27.2
a6–60	789.40*	789.40	6408	2746	13947	789.40	0.00	789.80	0.05	32.6
a6–72	910.24*	910.24	8799	2375	20200	910.24	0.00	910.24	0.00	47.6
a7–56	688.51*	688.51	1097	424	6625	688.51	0.00	688.51	0.00	27.8
a7–70	886.34	865.36	14401	1378	22931	887.53	2.56	889.75	2.82	39.4
a7–84	1015.11	1003.34[b]	14400	3369	23331	1014.12	0.89	1016.67	1.15	56.4
a8–64	713.11*	713.11	1331	735	6802	713.11	0.00	715.56	0.34	38.2
a8–80	925.56	904.73	14403	798	21223	925.56	2.30	926.63	2.42	48.0
a8–96	1196.35	1169.75	14404	834	8734	1196.79	2.31	1204.49	2.97	63.8
$\overline{I-MD}$		799.69	6288	1071	10929	806.42	0.67	808.01	0.84	38.7
$\overline{UEI-MD}$		803.47	3973	796	7673	806.67	0.31	807.94	0.45	40.1

[a] A slightly better lower bound was found by an earlier version of the algorithm: 1215.38.
[b] A slightly better lower bound was found by an earlier version of the algorithm: 1005.13.

instances. The algorithm outperforms state-of-the-art meta-heuristics for both the DARP and the H-DARP. Moreover, computation times are considerably smaller than for other algorithms, among others due to efficiently implementing feasibility checks and local search moves. Based on these findings, it is concluded that the branch-and-cut algorithm may be applied to solve small problem instances, while the meta-heuristic algorithm is an excellent and efficient tool for solving larger problems.

Future research might focus on adapting the proposed algorithms to consider alternative objective functions (e.g. penalize user inconvenience) or additional real-life problem characteristics (e.g. driver breaks, time-dependent travel times). Especially the introduction of time-dependent travel times in dial-a-ride problems seems a challenging yet highly relevant research direction since dial-a-ride services often take place in heavily congested urban areas. From the perspective of the static problem, the goal might be to develop vehicle schedules which take into account travel time variation during the day and which are robust to possible changes in these travel times. This topic has received little or no research attention so far. Finally, the multi-depot version of the DARP introduced in this paper enables opportunities for further research. In this paper the depots are considered to be at fixed locations nicely spread over the service region. Future research may analyze the relationship between depot locations and distance traveled. Additionally, opportunities for cooperation between service providers, offering dial-a-ride services in the same service area, may be analyzed.

Table 9
DARP: meta-heuristic results on 20 instances of Cordeau and Laporte (2003b).

Instance	BKS[a]	Hybrid LNS of Parragh and Schmid (2013)[b]					DA algorithm					New
		Best	Gap	Avg.	Gap	CPU	Best	Gap	Avg.	Gap	CPU	BKS
R1a	190.02	190.02	0.00	190.02	0.00	32.4	190.02	0.00	190.02	0.00	16.6	
R2a	301.34	301.34	0.00	302.53	0.40	165.6	301.34	0.00	301.34	0.00	42.0	
R3a	532.00	535.28	0.62	538.21	1.17	306.6	532.10	0.02	533.54	0.29	48.8	
R4a	570.25	571.09	0.15	576.26	1.05	977.4	577.16	1.21	580.52	1.80	74.6	
R5a	627.68	629.52	0.29	637.59	1.58	1602.0	629.80	0.34	632.06	0.70	89.2	626.93
R6a	785.26	788.88	0.46	800.35	1.92	2908.8	797.78	1.59	800.68	1.96	107.0	
R7a	291.71	291.71	0.00	292.56	0.29	61.2	292.23	0.18	292.23	0.18	22.6	
R8a	487.84	491.93	0.84	495.20	1.51	355.2	490.94	0.63	491.00	0.65	48.6	
R9a	658.31	661.47	0.48	676.09	2.70	1493.4	662.64	0.66	666.65	1.27	72.2	
R10a	855.15	872.31	2.01	878.93	2.78	2830.2	853.98	-0.14	860.83	0.66	114.4	851.82
R1b	164.46	164.46	0.00	166.06	0.97	36.6	164.46	0.00	164.46	0.00	23.8	
R2b	295.66	295.96	0.10	298.88	1.09	180.0	295.69	0.01	296.06	0.13	51.4	
R3b	484.83	484.83	0.00	491.29	1.33	491.4	488.61	0.78	490.03	1.07	76.2	
R4b	529.33	534.84	1.04	541.19	2.24	1354.8	534.99	1.07	540.99	2.20	117.0	
R5b	577.98	587.67	1.68	590.22	2.12	2645.4	581.46	0.60	584.33	1.10	155.2	577.29
R6b	737.69	738.01	0.04	743.64	0.81	4290.0	743.56	0.80	747.19	1.29	180.6	730.67
R7b	248.21	248.21	0.00	248.21	0.00	78.0	249.33	0.45	249.33	0.45	34.0	
R8b	461.39	463.67	0.49	470.25	1.92	572.4	461.77	0.08	462.38	0.21	81.0	458.73
R9b	593.49	593.49	0.00	606.25	2.15	1649.4	598.23	0.80	600.63	1.20	146.4	
R10b	793.21	804.22	1.39	812.81	2.47	4174.2	795.08	0.24	801.89	1.09	162.8	785.68
Avg	509.29	512.45	0.48	517.83	1.42	1310.3	512.06	0.47	514.31	0.81	83.2	

[a] Best known solutions, provided by Parragh and Schmid (2013).
[b] Results on 2.67 GHz Xeon computer with 24 GB RAM (shared with 7 other CPU's).

Table 10
DARP: meta-heuristic results on *b* instances of Cordeau (2006) and Ropke et al. (2007).

Instance	Optimal[a]	Hybrid LNS of Parragh and Schmid (2013)[b]					DA algorithm				
		Best	Gap	Avg.	Gap	CPU	Best	Gap	Avg.	Gap	CPU
b2–16	309.41	309.41	0.00	309.41	0.00	9.0	309.41	0.00	309.41	0.00	12.8
b2–20	332.64	332.64	0.00	332.64	0.00	12.6	332.64	0.00	332.64	0.00	10.0
b2–24	444.71	444.71	0.00	444.83	0.03	24.0	444.71	0.00	444.71	0.00	16.6
b3–24	394.51	394.51	0.00	394.51	0.00	18.6	394.51	0.00	394.51	0.00	13.2
b3–30	531.44	531.45	0.00	531.45	0.00	28.8	531.45	0.00	531.45	0.00	18.0
b3–36	603.79	603.79	0.00	604.13	0.06	44.4	603.79	0.00	603.79	0.00	20.2
b4–32	494.82	494.82	0.00	494.82	0.00	25.8	494.82	0.00	494.82	0.00	14.4
b4–40	656.63	656.63	0.00	656.63	0.00	60.0	656.63	0.00	656.63	0.00	26.8
b4–48	673.81	673.81	0.00	674.93	0.17	103.8	673.81	0.00	673.81	0.00	34.8
b5–40	613.72	613.72	0.00	613.73	0.00	46.8	613.72	0.00	613.72	0.00	22.6
b5–50	761.40	761.40	0.00	762.12	0.09	89.4	761.40	0.00	761.40	0.00	29.6
b5–60	902.04	902.52	0.05	903.46	0.16	180.0	902.04	0.00	902.04	0.00	41.4
b6–48	714.83	714.83	0.00	714.83	0.00	64.2	714.83	0.00	714.83	0.00	25.0
b6–60	860.07	860.07	0.00	860.15	0.01	124.2	860.07	0.00	860.07	0.00	35.6
b6–72	978.47	979.61	0.12	980.27	0.18	265.8	978.47	0.00	978.47	0.00	51.6
b7–56	823.97	823.97	0.00	824.17	0.02	109.2	823.97	0.00	823.97	0.00	31.8
b7–70	912.62	913.11	0.05	914.97	0.26	222.0	912.62	0.00	912.62	0.00	39.8
b7–84	1203.37	1211.27	0.66	1213.55	0.85	403.2	1203.81	0.04	1204.49	0.09	54.2
b8–64	839.89	840.63	0.09	840.63	0.09	153.6	839.89	0.00	839.89	0.00	34.2
b8–80	1036.34	1036.65	0.03	1038.28	0.19	230.4	1036.34	0.00	1036.34	0.00	47.6
b8–96	1185.55	1187.80	0.19	1190.74	0.44	619.2	1185.65	0.01	1185.88	0.03	63.2
Avg	727.33	727.97	0.06	728.58	0.12	135.0	727.36	0.00	727.40	0.01	30.6

[a] Optimal solutions provided by Ropke et al. (2007).
[b] Results on 2.67 GHz Xeon computer with 24 GB RAM (shared with 7 other CPU's).

Acknowledgments

This work is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (research project COMEX, Combinatorial Optimization: Metaheuristics & Exact Methods). The authors would like to thank the anonymous reviewers for their useful comments which helped to improve the quality of the paper.

Table A.1
Contribution of local search operators.

Excluded operator	Average gap	Average CPU
–	0.22	32.99
relocate	2.14	30.74
exchange	0.43	18.96
*2-opt**	0.50	35.08
r-4-opt	0.35	37.02

Appendix A. Meta-heuristic algorithm design decisions

As a meta-heuristic procedure, like deterministic annealing, may be applied in a variety of ways to a specific problem, several algorithmic configurations have been considered by the authors during the algorithm development phase. In Section 5, the algorithm configuration which yielded best results has been discussed in detail. The usefulness of the reduction parameter T_{red} and the restart mechanism was already demonstrated in Section 6.2. In this appendix, the contribution of each of the individual local search operators is analyzed and several alternative algorithm configurations are discussed.

To test the contribution to solution quality of a local search operator, the algorithm is re-run while excluding this operator from being used. Each time, five runs are performed on all DARP, H-DARP and MD-H-DARP instances used in Sections 6.3, 6.4, 6.5 (149 in total). Results for the four distance-related local search operators (*relocate*, *exchange*, *2-opt**, *r-4-opt*) are given in Table A.1. Average gaps and average computation times in seconds are reported. Results in Table A.1 indicate that each of the operators contributes significantly to solution quality and therefore should be part of the algorithm. Especially the importance of the *relocate* operator stands out.

When the *eliminate* operator is excluded, the algorithm is unable to find a feasible solution in 32 out of 745 runs (4.3%). For all other runs, average gaps are the same as when the operator is applied (0.22%) although the algorithm becomes slower (40.46 s compared to 31.57 s). The *eliminate* operator may therefore be regarded an essential part of the algorithm as well (with the main purpose of finding a feasible solution).

A similar analysis is performed with respect to the contribution of the insertion heuristic. Results of the proposed algorithm are compared with results obtained when only a single iteration of the insertion heuristic is used (inserting user requests in increasing order of the start of the time window at the pickup vertex). Results are slightly worse on average (0.28% compared to 0.22%), indicating that the meta-heuristic algorithm benefits from having a good initial solution. However, a more careful analysis of the results indicates that this difference in solution quality is mainly due to four small instances (I-a2–20, I-a2–24, I-MD-a2–20, I-MD-a2–24). When using only a single iteration of the insertion heuristic on these instances, the algorithm tends to get stuck in a local optimum and does not succeed in finding the optimal solution due to the limited number of feasible local search moves available from the starting solution. However, when applying the insertion heuristic a thousand times, as in the proposed algorithm, the optimal solution is already obtained by the insertion heuristic. For all other instances, results of using a single or a thousand iterations of the insertion heuristic are very similar (average gaps of 0.24% and 0.23% respectively), although computation times are slightly larger when using only a single iteration (35.77 s compared to 33.16 s). This is due to the fact that the insertion heuristic is very fast and starting the meta-heuristic from a better initial solution reduces the number of profitable local search moves which require evaluation of feasibility.

In Table A.2, results of the proposed meta-heuristic are compared with results of several alternative algorithm configurations. First, the effect of using the *eliminate* operator in every iteration is tested. In the proposed algorithm, the operator is only used when the current number of vehicles is larger than the amount available. Results indicate that when the operator is used in every iteration (accepting solutions according to the threshold value T when $k(\boldsymbol{x}) \leqslant n^v$), solution quality hardly changes while computation times increase drastically. Therefore, only using the *eliminate* operator to reduce the number of vehicle when $k(\boldsymbol{x}) > n^v$ seems to be the best option.

For the intra-route operator, the effect of using three or five successive arcs instead of four is analyzed. With respect to using only three successive arcs (*r-3-opt*), this yields an operator in which the only option is to reverse the visiting order of two successive vertices. Results are worse compared to using four successive arcs. With respect to using five successive arcs

Table A.2
Results of alternative algorithm configurations.

Algorithm	Average gap	Average CPU
Base	0.22	32.99
Always use *eliminate*	0.23	71.52
r-3-opt instead of *r-4-opt*	0.26	35.26
r-5-opt instead of *r-4-opt*	0.22	45.75
Two-phase algorithm	0.35	28.98
Introduce tabu search (θ = 10)	0.32	35.21
Introduce tabu search (θ = 20)	0.46	33.10

(*r-5-opt*), hardly any effect on solution quality is detected, although computation time increases considerably due to the increased number of possible orderings to evaluate at each iteration. A similar effect is expected when applying the operator to even longer segments of routes. Hence, using four successive arcs is the best choice.

Finally, two more complex configurations of the deterministic annealing algorithm are considered: a two-phase version of the algorithm and the introduction of short term memory. Both approaches have been proven successful for a bi-objective vehicle routing problem in the context of drayage operations (Braekers et al., 2014). The two-phase version of the algorithm is as follows. In the first phase, minimizing the number of vehicles is prioritized over minimizing total distance. For this purpose, the *eliminate* operator is applied in every iteration and solutions which use less vehicles are always accepted. In the second phase, iteratively total distance is minimized (while keeping the number of vehicles constant) and the number of vehicles is increased by one, until the maximum number of vehicles is reached. The idea is that this approach offers a more profound search for minimal distance solutions for each possible number of vehicles used. A similar sensitivity analysis on the algorithm parameters was performed as in Section 6.2. However, results indicate that this approach does not offer the same solution quality as the algorithm in Section 5. Finally, the introduction of tabu search logic (short term memory) is tested. Arcs which are removed from a solution are not allowed to re-enter the solution for θ iterations in order to prevent cycling between solutions. However, results become worse when θ is increased. Hence, the alternative and more complex algorithm configurations do not provide an improvement.

References

Ascheuer, N., Fischetti, M., Grotschel, M., 2000. A polyhedral study of the asymmetric traveling salesman problem with time windows. Networks 36 (2), 69–79.

Atahran, A., Lenté, C., T'kindt, V., 2014. A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. J. Multi-Criteria Decis. Anal. (forthcoming). http://dx.doi.org/10.1002/mcda.1518.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. Eur. J. Oper. Res. 202 (1), 8–15.

Berbeglia, G., Pesant, G., Rousseau, L.-M., 2011. Checking the feasibility of dial-a-ride instances using constraint programming. Transp. Sci. 45 (3), 399–412.

Braekers, K., Caris, A., Janssens, G.K., 2013. Integrated planning of loaded and empty container movements. OR Spectrum 35 (2), 457–478.

Braekers, K., Caris, A., Janssens, G.K., 2014. Bi-objective optimization of drayage operations in the service area of intermodal terminals. Transp. Res. Part E: Logis. Transp. Rev. 65, 50–69.

Bräysy, O., Berger, J., Barkaoui, M., Dullaert, W., 2003. A threshold accepting metaheuristic for the vehicle routing problem with time windows. Cent. Eur. J. Oper. Res. 11 (4), 369–387.

Bräysy, O., Dullaert, W., Hasle, G., Mester, D., Gendreau, M., 2008. An effective multistart deterministic annealing metaheuristic for the fleet size and mix vehicle routing problem with time windows. Transp. Sci. 42 (3), 371–386.

Bräysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, part II: metaheuristics. Transp. Sci. 39 (1), 119–139.

Campbell, A.M., Savelsberg, M., 2004. Efficient insertion heuristics for vehicle routing and scheduling problems. Transp. Sci. 38 (3), 369–378.

Caris, A., Janssens, G.K., 2010. A deterministic annealing algorithm for the pre- and end-haulage of intermodal container terminals. Int. J. Comput. Aided Eng. Technol. 2 (4), 340–355.

Carnes, T.A., Henderson, S.G., Shmoys, D.B., Ahghari, M., MacDonald, R.D., 2013. Mathematical programming guides air-ambulance routing at ornge. Interfaces 43 (3), 232–239.

Chevrier, R., Liefooghe, A., Jourdan, L., Dhaenens, C., 2012. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: application to demand responsive transport. Appl. Soft Comput. 12 (4), 1247–1258.

Cordeau, J.-F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Oper. Res. 54 (3), 573–586.

Cordeau, J.-F., Laporte, G., 2003a. The Dial-a-Ride Problem (DARP): variants, modeling issues and algorithms. 4OR: Quart. J. Oper. Res. 1 (2), 89–101.

Cordeau, J.-F., Laporte, G., 2003b. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B: Methodol. 37 (6), 579–594.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Ann. Oper. Res. 153 (1), 29–46.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., Savelsbergh, M.W.P., 2007a. Transportation on demand. In: Barnhart, C., Laporte, G. (Eds.), Transportation, Handbooks in Operations Research and Management Science, vol. 14. Elsevier, pp. 429–466.

Cordeau, J.-F., Laporte, G., Savelsbergh, M.W.P., Vigo, D., 2007b. Vehicle routing. In: Barnhart, C., Laporte, G. (Eds.), Transportation, Handbooks in Operations Research and Management Science, vol. 14. Elsevier, pp. 367–428.

Diana, M., Dessouky, M.M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transp. Res. Part B: Methodol. 38 (6), 539–557.

Diana, M., Dessouky, M.M., Xia, N., 2006. A model for the fleet sizing of demand responsive transportation services with time windows. Transp. Res. Part B: Methodol. 40 (8), 651–666.

Dueck, G., Scheuer, T., 1990. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J. Computat. Phys. 90 (1), 161–175.

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. Eur. J. Oper. Res. 54 (1), 7–22.

Gendreau, M., Potvin, J.-Y., Bräysy, O., Hasle, G., Løkketangen, A., 2008. Metaheuristics for the vehicle routing problem and its extensions: a categorized bibliography. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, Operations Research/Computer Science Interfaces, vol. 43. Springer, US, pp. 143–169.

Guerriero, F., Bruni, M.E., Greco, F., 2013. A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem. Asia-Pac. J. Oper. Res. 30 (1), 1250046.

Häme, L., 2011. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows. Eur. J. Oper. Res. 209 (1), 11–22.

Häme, L., Hakula, H., 2013. Routing by ranking: a link analysis method for the constrained dial-a-ride problem. Oper. Res. Lett. 41 (6), 664–669.

Hunsaker, B., Savelsbergh, M., 2002. Efficient feasibility testing for dial-a-ride problems. Oper. Res. Lett. 30 (3), 169–173.

Jain, S., Van Hentenryck, P., 2011. Large neighborhood search for dial-a-ride problems. In: Lee, J. (Ed.), Principles and Practice of Constraint Programming CP 2011, Lecture Notes in Computer Science, 6876. Springer, Berlin Heidelberg, pp. 400–413.

Jørgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2007. Solving the dial-a-ride problem using genetic algorithms. J. Oper. Res. Soc. 58 (10), 1321–1331.

Kindervater, G.A.P., Savelsbergh, M.W.P., 1997. Vehicle routing: handling edge exchanges. In: Aarts, E., Lenstra, J.K. (Eds.), Local Search in Combinatorial Optimization. Wiley, pp. 337–360.

Kirchler, D., Wolfler Calvo, R., 2013. A granular tabu search algorithm for the dial-a-ride problem. Transp. Res.Part B: Methodol. 56, 120–135.

Lehuédé, F., Masson, R., Parragh, S.N., Péton, O., Tricoire, F., 2014. A multi-criteria large neighbourhood search for the transportation of disabled people. J. Oper. Res. Soc. (forthcoming). http://dx.doi.org/10.1057/jors.2013.17.

Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21 (2), 498–516.

Luo, Y., Schonfeld, P., 2007. A rejected-reinstertion heuristic for the static dial-a-ride problem. Transp. Res.Part B: Methodol. 41 (7), 736–755.

Melachrinoudis, E., Ilhan, A.B., Min, H., 2007. A dial-a-ride problem for client transportation in a health-care organization. Comput. Oper. Res. 34 (3), 742–759.

Mitrović-Minić, S., 1998. Pickup and delivery problem with time windows: a survey. Technical Report SFU CMPT TR 1998–12, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

Naddef, D., Rinaldi, G., 2002. Branch-and-cut algorithms for the capacitated vrp. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, pp. 53–84.

Neven, A., Braekers, K., Declerq, K., Bellemans, T., Janssens, D., Wets, G., 2014. Methodology to optimize resource requirements of a demand responsive transport system for persons with disabilities: a case study on Flanders. In: 93th Annual Meeting of the Transportation Research Board. Transportation Research Board, Washington, DC.

Nikolakopoulos, A., Sarimveis, H., 2007. A threshold accepting heuristic with intense local search for the solution of special instances of the traveling salesman problem. Eur. J. Oper. Res. 177 (3), 1911–1929.

Paquette, J., Cordeau, J.-F., Laporte, G., Pascoal, M.M.B., 2013. Combining multicriteria analysis and tabu search for dial-a-ride problems. Transp. Res. Part B: Methodol. 52, 1–16.

Parragh, S.N., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transp. Res. Part C: Emerg. Technol. 19 (5), 912–930.

Parragh, S.N., Cordeau, J.-F., Doerner, K.F., Hartl, R.F., 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. OR Spectrum 34 (3), 593–633.

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008. A survey on pickup and delivery problems. Part II: transportation between pickup and delivery locations. J. Betriebswirtschaft 58 (2), 81–117.

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 37 (6), 1129–1138.

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

Parragh, S.N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 40 (1), 490–497.

Potvin, J.-Y., Rousseau, J.-M., 1995. An exchange heuristic for routeing problems with time windows. J. Oper. Res. Soc. 46 (12), 1433–1446.

Qu, Y., Bard, J.F., 2013. The heterogeneous pickup and delivery problem with configurable vehicle capacity. Transp. Res. Part C: Emerg. Technol. 31, 1–20.

Rekiek, B., Delchambre, A., Saleh, H.A., 2006. Handicapped person transportation: an application of the grouping genetic algorithm. Eng. Appl. Artif. Intell. 19 (5), 511–520.

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

Savelsbergh, M.W.P., 1992. The vehicle routing problem with time windows: Minimizing route duration. ORSA J. Comput. 4 (2), 146–154.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Eur. J. Oper. Res. 152 (1), 148–158.

Wolfler Calvo, R., Colorni, A., 2007. An effective and fast heuristic for the dial-a-ride problem. 4OR: Quart. J. Oper. Res. 5 (1), 61–73.

Wolfler Calvo, R., Touati-Moungla, N., 2011. A matheuristic for the dial-a-ride problem. In: Pahl, J., Reiners, T., Vo, S. (Eds.), Network Optimization, Lecture Notes in Computer Science, 6701. Springer, Berlin Heidelberg, pp. 450–463.

Wong, K.I., Bell, M.G.H., 2006. Solution of the dial-a-ride problem with multi-dimensional capacity constraints. Int. Trans. Oper. Res. 13 (3), 195–208.

Xiang, Z., Chu, C., Chen, H., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. Eur. J. Oper. Res. 174 (2), 1117–1139.