European Journal of Operational Research 209 (2011) 11–22

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows

Lauri Häme *

Aalto University School of Science and Technology, PL 11000, 00076 Aalto, Finland

ARTICLE INFO

Article history:
Received 29 January 2010
Accepted 24 August 2010
Available online 27 August 2010

Keywords:
Transportation
Dial-a-ride problem
Exact algorithm
Heuristics

ABSTRACT

The dial-a-ride problem (DARP) is a widely studied theoretical challenge related to dispatching vehicles in demand-responsive transport services, in which customers contact a vehicle operator requesting to be carried from specified origins to specified destinations. An important subproblem arising in dynamic dial-a-ride services can be identified as the single-vehicle DARP, in which the goal is to determine the optimal route for a single vehicle with respect to a generalized objective function. The main result of this work is an adaptive insertion algorithm capable of producing optimal solutions for a time constrained version of this problem, which was first studied by Psaraftis in the early 1980s. The complexity of the algorithm is analyzed and evaluated by means of computational experiments, implying that a significant advantage of the proposed method can be identified as the possibility of controlling computational work smoothly, making the algorithm applicable to any problem size.

1. Introduction

Dial-a-ride involves the dispatching of a fleet of vehicles to satisfy demands from customers who contact a vehicle operating agency requesting to be carried from specified origins to other similarly specified destinations, as defined in Psaraftis (1980).

Different types of dial-a-ride services give rise to different types of mechanisms for controlling vehicle operations. For example, if a dial-a-ride service requires customers to request service during the previous day, the nature of vehicle dispatching will certainly differ from a service in which customers may request immediate service. The problem of dispatching vehicles in a dial-a-ride service is generally called the *dial-a-ride problem* (DARP). In this work, a specific version of this problem is examined, namely, the *single-vehicle DARP with time windows*, in which the goal is to determine the optimal route for a single vehicle serving a certain set of customers.

The consideration of narrow time windows means that the vehicle route is restricted by relatively strict time limits for pickup and delivery of each customer. Narrow time windows emerge in real-time dial-a-ride services, in which each customer is given an estimate or guarantee regarding the pickup and delivery times in the form of time windows. These time windows are examined as hard limits to be met by the vehicle. Time windows have been incorporated in many early and recent studies of the DARP (see for example Psaraftis, 1983; Jaw et al., 1986; Madsen et al., 1995; Toth and Vigo, 1997; Cordeau and Laporte, 2003a; Diana and Dessouky, 2004; Wong and Bell, 2006; Cordeau, 2006). In these studies it is noted that in dynamic settings, time windows eliminate the possibility of indefinite deferment of customers and strict time limits help provide reliable service. While the main focus of this work is on fixed time windows, maximum ride time constraints that limit the time spent by passengers on the vehicle between their pickup and their delivery (Hunsaker and Savelsbergh, 2002), are considered as well.

The problem formulation studied in this work follows closely the one presented in Psaraftis (1980, 1983). In the first reference, the objective function is defined as a generalization of the objective function of the Traveling Salesman Problem (TSP), in which a weighted combination of the time needed to serve all customers and of the total degree of dissatisfaction they experience until their arrival to the destination of the trip is minimized. The dissatisfaction of customers is assumed to be a linear function of the time each customer waits to be picked up and the time each customer spends riding in the vehicle until his/her delivery. In the second reference, the approach is extended to handle time windows on departure and arrival times, but only the route duration is minimized.

In this work, both aspects of the problem (general objective function and time windows) are considered. The main contribution is a solution method designed in a way that (i) it is capable of handling practically any objective function suitable for dynamic routing and (ii) the computational effort of the algorithm can be controlled smoothly: if the problem size is reasonable, the algorithm produces optimal solutions efficiently and as the problem size is increased, the search space may be narrowed in order to achieve locally optimal solutions.

* Tel.: +358 40 576 3585; fax: +358 9 451 2474.
E-mail address: lauri.hame@tkk.fi

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.08.021

This document is organized as follows. Section 2 introduces an exact algorithm for the static version of the problem, in which all requests are known a priori. As the complexity of the problem is increased, the algorithm is extended to a heuristic procedure with slight adjustments in order to be able to solve the problem more efficiently (Section 2.3). This is followed by a discussion of the dynamic case in Section 3 and by computational results presented in Section 4, where the complexity and performance of the proposed solution method is evaluated.

1.1. Literature review

As mentioned above, different versions of the dial-a-ride problem motivate several different solution methods. Most recent studies are related to the static multiple vehicle DARP, in which a set of vehicle routes is designed for a set of customers, whose pickup and drop-off points are known a priori (see for example Cordeau and Laporte, 2003a; Cordeau, 2006; Bent and Van Hentenryck, 2006; Ropke and Pisinger, 2006; Xiang et al., 2006; Melachrinoudis et al., 2007; Parragh et al., 2010; Garaix et al., 2010). In addition, a few heuristics for the dynamic version have recently been reported (for example Madsen et al., 1995; Horn, 2002; Attanasio et al., 2004; Coslovich et al., 2006; Xiang et al., 2008).

In the following paragraphs, a short review on the studies related to the single-vehicle dial-a-ride problem is presented, based on Cordeau and Laporte (2003b), Cordeau et al. (2007), Cordeau and Laporte (2007) and Berbeglia et al. (2010), to which the reader is referred for more exhaustive summaries.

The first exact approach to the solution of the dynamic dial-a-ride problem was presented by Psaraftis (1980). In this work the single-vehicle, "immediate-request" case is studied, in which a set of customers should be served as soon as possible. In this first model no time windows are specified by the customers, but the vehicle operator incorporates *maximum position shift* constraints limiting the difference between the position of a request in the chronological calling list and its position in the vehicle route. The objective is to minimize a weighted combination of the time needed to serve all customers and customer dissatisfaction. The complexity of this algorithm is $\mathcal{O}(n^2 3^n)$ and only small instances can thus be solved. However, the computational effort is decreased as more strict constraints are imposed.

The algorithm is first constructed for the static case of the problem and then extended to solve the dynamic case, in which new requests occur during the execution of the route but no information on future requests is available. In this version of the problem, the use of maximum position shift constraints is essential, in order to prevent a request from being indefinitely deferred. In a later study (Psaraftis, 1983), the approach is extended to handle time windows on departure and arrival times. In this extension, only the route duration is considered in the objective function.

In Sexton and Bodin (1985a,b), a heuristic approach to the single-vehicle DARP with one-sided time windows was introduced. In this problem formulation, the objective function is defined as a function of (i) the difference between the actual travel time and the direct travel time of a user and (ii) the difference between desired drop-off time and actual drop-off time. The algorithm was tested on real-life problems involving 7 to 20 customers.

In Desrosiers et al. (1986), an exact dynamic programming algorithm for the single-vehicle DARP, formulated as an integer program, was presented. The formulation includes time windows as well as vehicle capacity and precedence constraints. Optimal solutions with respect to total route length were obtained for $n = 40$.

In Bianco et al. (1994), exact and heuristic procedures for the traveling salesman problem with precedence constraints are introduced, based on a bounding procedure. Computational results of the algorithm are given for a number of randomly generated test problems, including the dial-a-ride problem with the classical TSP objective function.

Psaraftis noted that although single-vehicle Dial-A-Ride systems do not exist in practice, single-vehicle Dial-A-Ride algorithms can be used as subroutines in large scale multi vehicle Dial-A-Ride environments. It is mainly for this reason that one's ability to solve the single-vehicle DARP is considered important. A similar approach is used in this work, where the idea is to use the solution of the single-vehicle DARP as a subroutine in a dynamic multiple-vehicle scenario. An effort is made to solve the single-vehicle problem up to optimality whenever the problem size is reasonable.

2. The static dial-a-ride problem with time windows

In this case of the dial-a-ride problem, it is assumed that there are N customers to be served, to each of which is associated a pickup point, a drop-off point and relatively narrow time windows for pickup and delivery. The problem is to determine for a single vehicle the optimal route with respect to the N customers at some instant ($t = 0$).

Since the main application of the algorithm is related to dynamic dial-a-ride services, vehicle routes will be examined as *open paths*, starting with the location of the vehicle at $t = 0$ and ending with the delivery of a customer, instead of round trips, as in most static vehicle routing and dial-a-ride problems. The use of open paths causes no loss of generality, as the open and closed route problems are reducible to one another in linear time. This has been shown for the Traveling Salesman Problem by Papadimitriou (1977).

In addition, the computational effort of the algorithm should be minimal in order to be able to produce optimal solutions in a short time, even though the number of customers is large. In Psaraftis (1980, 1983), an exact algorithm based on dynamic programming is introduced for the single-vehicle DARP similar to the one described above. It is shown that if the problem is solved with strict constraints, the algorithm proves to solve the problem to optimality in short CPU times. However, if the constraints are not restrictive, the computational effort grows as an exponential function of the number of customers N. In the approach presented in this work, an effort is made to solve the problem up to optimality whenever the computational effort is reasonable. If the computational effort needed to produce globally optimal solutions grows too large, the algorithm should still produce locally optimal solutions with relatively little computational effort. In other words, the algorithm should be scalable in a way that the computing time may be somehow controlled regardless of the number of customers.

2.1. Problem formulation (Psaraftis, 1980, 1983)

Let there be N customers, each of which have been assigned a number i between 1 and N. For each customer $i \in \{1,\ldots,N\}$, let u_i be the pickup point and u_{N+i} be the delivery point, $[e_i, l_i]$ be the pickup time window and $[e_{N+i}, l_{N+i}]$ be the delivery time window,[1] q_i be the load (number of passengers) associated to customer i. Let A be the starting point of the vehicle (location of the vehicle at $t = 0$). It is assumed that the time to go from any one of the pickup and delivery points u_i, where $i \in \{1,\ldots,2N\}$, directly to another point u_j is a known and fixed quantity $t(i,j)$.

The goal is to find a vehicle route starting from A and ending at one of the delivery points so that the following conditions hold.

[1] In addition to the fixed time windows $[e_i, l_i]$ and $[e_{N+i}, l_{N+i}]$, to each customer may be associated a specific maximum ride time constraint $T_i^{\max}$. The handling of these constraints is discussed in Section 2.2.1.

1. The quantity

$$w_1 T + w_2 \sum_{i=1}^{N} \left(\alpha T_i^W + (1-\alpha) T_i^R \right) \quad (1)$$

is minimized, where
   - $w_1, w_2 \in \mathbb{R}$ are given weight parameters,
   - the parameter α is the customers' time preference constant ($0 \leqslant \alpha \leqslant 1$),
   - T is the duration of the route,
   - T_i^W is the waiting time of customer i, from the earliest pickup time until the time of pickup,
   - T_i^R is the riding time of customer i, from the time of pickup until the time of delivery.
2. The vehicle route should be legitimate, namely each customer should be picked up before he/she is delivered.
3. The vehicle has a certain capacity of C passengers that cannot be exceeded.
4. All time constraints must be satisfied.

In Eq. (1), the quantity $\sum_{i=1}^{N} \left(\alpha T_i^W + (1-\alpha) T_i^R \right)$ is the assumed form of the total degree of dissatisfaction experienced by the customers until their delivery, where α is the customers' time preference constant describing the impact of ride time and waiting time on the degree of dissatisfaction.

In Powell et al. (1995), it is noted that in static and deterministic transportation models, finding an appropriate objective function is fairly easy and that the objective function is usually a good measure for evaluating the solution. In dynamic models, the objective function used to find the solution over a rolling horizon has often little to do with the measures developed to evaluate the overall quality of a solution. In stochastic models it might be useful to minimize the probability of violating time windows. However, the advanced insertion method to be described is designed in a way that several objective functions, that are thought to be suitable for dynamic and stochastic problems, may be incorporated with minimal work. Generally, the choice of the objective function depends on the particular version of the problem and is discussed only briefly in this document. For a study related to choosing an appropriate objective function for the dynamic DARP, the reader is referred to Hyytiä et al. (2010).

The feasibility of time windows is governed by the following two assumptions.

1. If the vehicle arrives at any node (either pickup or delivery) later than the upper bound of that node's time constraint (l_j), then the entire vehicle route and schedule is infeasible. In other words, those upper bounds constitute hard constraints that should be met by the vehicle.
2. If the vehicle arrives at any node earlier than the lower bound on that node's time constraint (e_j), then the vehicle will stay idle at that point and depart immediately at e_j.

The algorithm to be presented in the following section will check that the time constraints are satisfied in a fashion particularly suitable for a highly restricted problem. In addition, the vehicle route is constructed in such a way that the pickup of each customer precedes the delivery and thus there is no need to check the validity of the legitimacy condition. While capacity constraints are also taken into account, the algorithm is designed and works best on problems in which the time limits are assumed to be more restrictive.

No exact solution to the problem described above has been reported. Similar approaches include the following.

- Psaraftis (1980), Bianco et al. (1994), Hernández-Pérez and Salazar-González (2009). Time limits are substituted by maximum position shift constraints.
- Psaraftis (1983), Desrosiers et al. (1986). Time windows are considered but the objective is to minimize route duration.
- Sexton (1979), Sexton and Bodin (1985a,b). A heuristic approach to a similar problem with one-sided time windows is presented.

2.2. Static case solution

In general, exact procedures for solving routing problems are computationally very taxing, since the complexity is always more or less equal to the classical Traveling Salesman Problem. In addition, exact optimization can be seen to be needless at run-time if routes are modified often. Despite these facts, exact algorithms are useful in a way that the performance of different heuristics may be compared to the optimal solution.

The main idea in the following approach to the solution of the static case is that customers are added to the vehicle route one by one by using an *advanced insertion* method, which leads to a globally optimal solution, that is a vehicle route which is feasible with respect to all customers and minimizes a given cost function.

Many studies related to the dial-a-ride problem (see for example Jaw et al., 1986; Madsen et al., 1995; Diana and Dessouky, 2004; Wong and Bell, 2006), make use of what is called the insertion procedure, in the classical version of which the pickup and delivery node of a new customer are inserted into the *current optimal sequence* of pickup and delivery nodes of existing customers. In these references, several improvements to the insertion algorithm have been suggested. However, the main idea of the classical insertion algorithm a priori excludes the possibility that the appearance of a new customer may render the optimal sequencing of already existing customers no longer optimal. While the basic insertion algorithm can be seen to produce relatively good results for the unconstrained TSP, the performance compared to an exact algorithm is decreased as the problem becomes more constrained (see Section 4).

In this work, the main idea is to construct the optimal route iteratively by implementing an insertion algorithm for each customer, one by one *for all feasible sequences of pickup and delivery nodes of existing customers*. Namely, the procedure involves two steps for each customer:

1. Perform insertion of the new customer to all feasible service sequences with respect to existing customers.
2. Determine the set of feasible service sequences with respect to the new customer and existing customers.

It can be readily shown that the insertion of a new customer to all feasible service sequences with respect to existing customers produces all feasible service sequences with respect to the union

Table 1
Potential service sequences with respect to customers 1 and 2. No capacity or time constraints are taken into account. $i^{\uparrow}$ denotes the pickup node and $i^{\downarrow}$ denotes the delivery node of customer i.

A:	$1^{\uparrow}$	$1^{\downarrow}$	$2^{\uparrow}$	$2^{\downarrow}$	B:	$1^{\uparrow}$	$2^{\uparrow}$	$1^{\downarrow}$	$2^{\downarrow}$	C:	$1^{\uparrow}$	$2^{\uparrow}$	$2^{\downarrow}$	$1^{\downarrow}$
D:	$2^{\uparrow}$	$1^{\uparrow}$	$1^{\downarrow}$	$2^{\downarrow}$	E:	$2^{\uparrow}$	$1^{\uparrow}$	$2^{\downarrow}$	$1^{\downarrow}$	F:	$2^{\uparrow}$	$2^{\downarrow}$	$1^{\uparrow}$	$1^{\downarrow}$

of existing customers and the new customer and leads to a globally optimal solution but is computationally expensive if the number of feasible service sequences grows large. However, in the following algorithm the route is constructed under relatively narrow time window constraints, which means that the number of feasible routes with respect to all customers will be small compared to the number of all legitimate routes. Furthermore, the algorithm may be easily extended to an adjustable heuristic algorithm able to handle any types of time windows, as will be seen in Section 2.3.

The idea of the advanced insertion method is clarified by the following example, where no capacity or time constraints are taken into account. Let $i^{\uparrow} = i$ denote the *pickup node* of customer i and let $i^{\downarrow} = N + i$ denote the *delivery node* of customer i. A *service sequence* is defined as an ordered list consisting of pickup and delivery nodes. For instance, the service sequence $(i^{\uparrow}, j^{\uparrow}, j^{\downarrow}, i^{\downarrow})$ indicates the order in which customers i and j are picked up and dropped off.

Let us start the advanced insertion process with customer 1. Since the pickup $1^{\uparrow}$ of customer 1 has to be before the delivery, $1^{\downarrow}$, the only possible service sequence at this point is $(1^{\uparrow}, 1^{\downarrow})$. Thus, the set of potential service sequences with respect to customer 1 consists of this single service sequence. By insertion of customer 2 into the service sequence $(1^{\uparrow}, 1^{\downarrow})$ we get the six service sequences presented in Table 1.

By inserting the pickup and delivery node of customer 3 into all of these service sequences we get a total of $6(5 + 4 + 3 + 2 + 1) = 90$ new potential service sequences. However, if the time and capacity constraints are taken into account, not all service sequences described above are necessarily feasible.

For example, if after the insertion of customer 2 it can be seen that only the service sequences A and B in Table 1, namely $(1^{\uparrow}, 1^{\downarrow}, 2^{\uparrow}, 2^{\downarrow})$ and $(1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow})$, are feasible with respect to time constraints, then it can be shown that all feasible service sequences including customer 3 are included in the sequences given by insertion to these two service sequences, which gives a maximum of $2(5 + 4 + 3 + 2 + 1) = 30$ new potential service sequences.

To formalize the above procedure, an insertion function is defined as follows. Let s be a service sequence $s = (p_1, p_2, \ldots, p_m)$ consisting of m nodes, where $p_j \in \{1, \ldots, 2N\}$ for all $j \in \{1, \ldots, m\}$ and let h and k be natural numbers such that $h \in \{1, \ldots, m + 1\}$ and $k \in \{h + 1, \ldots, m + 2\}$. The insertion $I(s, i, h, k)$ of a new customer i into the sequence s is defined as the service sequence

$$I(s,i,h,k) = (r_1, \ldots, r_{m+2}) = (p_1, p_2, \ldots, p_{h-1}, i^{\uparrow}, p_h, \ldots, p_{k-2}, i^{\downarrow}, p_{k-1}, \ldots, p_m). \quad (2)$$

For example, an insertion $I(s,i,h,k) = I((1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow}), 3, 2, 4)$ would produce the service sequence $(1^{\uparrow}, 3^{\uparrow}, 2^{\uparrow}, 3^{\downarrow}, 1^{\downarrow}, 2^{\downarrow})$. In other words, the numbers $h = 2$ and $k = 4$ mark the positions of pickup and delivery of the new customer 3 in the new service sequence. Furthermore, the insertion $I(\emptyset, 1, 1, 2)$ would correspond to the service sequence $(1^{\uparrow}, 1^{\downarrow})$.

By means of the above definition, we can now describe the structure of the advanced insertion algorithm specifically, see Algorithm 1. Briefly, the structure of the advanced insertion algorithm may be sketched as follows. At first, the set of feasible service sequences S_i is determined recursively for each customer $i \in \{1, \ldots, N\}$ by inserting the pickup and delivery nodes of customer i into each feasible service sequence with respect to customers 1, …, $i - 1$. In this way the algorithm produces the set S_N of all feasible routes with respect to customers 1, …, N. Then the solution to the static problem is obtained by choosing the sequence $s \in S_N$ with minimal cost $C(s)$. In the general form (1) of the cost function, this involves the calculation of waiting times and ride times for all customers for all feasible service sequences. If only route duration is minimized, we can simply choose the sequence for which the arrival time at the last node is smallest.

Algorithm 1: The advanced insertion algorithm

```
Set S_0 = {∅};        (S_i = set of feasible service sequences with
   respect to customers 1, ..., i)
for each i ∈ {1,...,N} do
  Set S_i = ∅;
  for each s ∈ S_{i-1} do
    for each h ∈ {1,...,|s| + 1} and k ∈ {h + 1, ..., |s| + 2} do
      Set r = I(s,i,h,k);      (I = the insertion function)
      if service sequence r is feasible then
        S_i = S_i ∪ {r};
      end if
    end for
  end for
  if S_i = ∅ then
    STOP: The problem has no solution;
  end if
endfor
for each s ∈ S_N
    Calculate cost C(s);
end for
```

In the following part, the verification of feasibility of sequences is examined.

2.2.1. Feasibility considerations

At each insertion, the feasibility of the new sequence $I(s,i,h,k) = (r_1, \ldots, r_m)$, where $s = (p_1, \ldots, p_{m-2})$, is verified. Since the precedence constraint is taken care of automatically in the insertion procedure, it is sufficient to check feasibility with respect to time and capacity constraints.

Time windows. The arrival time t_j at the jth node of the route should satisfy $t_j \leqslant l_{r_j}$, where l_{r_j} is the latest arrival time of node r_j and t_j is defined recursively by

$$t_j = \max(t_{j-1}, e_{r_{j-1}}) + t(r_{j-1}, r_j), \quad (3)$$

where $t_0 = 0$. For simplicity, the time needed for each customer to get on the vehicle or get off the vehicle is not taken into account in this study. However, unique service times $d_i, d_{N+i} \in \mathbb{R}$ for each customer's pickup and delivery may be incorporated in the model with minimal effort.

Maximum ride time constraints. If maximum ride time constraints are considered in addition to the fixed time windows, the feasibility is checked as follows. Let $T_i^{\max}$ denote the maximum ride time of customer i. When the arrival time t_j at pickup node r_j is calculated by means of formula (3), the latest delivery time of customer r_j is updated by means of the formula

$$l'_{N+r_j} = \min\left(l_{N+r_j}, \max(t_j, e_{r_j}) + T_{r_j}^{\max}\right), \quad (4)$$

where $\max(t_j, e_{r_j})$ denotes the pickup time of customer r_j. The arrival time t_k at each delivery node r_k should then satisfy $t_k \leqslant l'_{r_k}$.

Capacity. The total load on the vehicle Y_j just after it leaves the jth node of the route should satisfy $Y_j \leqslant C$, where C is the capacity of the vehicle and Y_j is defined recursively by

$$Y_j = Y_{j-1} + q_{r_j}, \quad (5)$$

where $Y_0 = 0$ and q_{r_j} is the load on node r_j.

The verification can be executed by means of the following procedure.

1. Set $t_j = \max\{t_{j-1}, e_{r_{j-1}}\} + t(j-1, r_j)$. If $t_j > l_{r_j}$, STOP: insertion $I(s,i,h,k)$ is infeasible.
2. If $j < k$: Set $Y_j = Y_{j-1} + q_{r_j}$. If $Y_j > C$, STOP: insertion $I(s,i,h,k)$ is infeasible.

If the insertion $r = I(s,i,h,k)$ passes the above procedure, the sequence $(r_1,\ldots,r_m)$ is added to the set of feasible service sequences S_i for the new customer i.

2.2.2. A priori clustering

In problems involving a large number of customers, assuming that the time windows are relatively narrow, a significant portion of service sequences can be eliminated before the actual insertion process by simply studying the mutual relationships between nodes, similarly as described in Dumas et al. (1991). More precisely, assume that the vehicle departs from a node i at the lower bound e_i of the time window and moves directly to node j. If the vehicle does make it in time to j, that is, if

$$e_i + t(i,j) > l_j, \tag{6}$$

it is said that the transition $i \to j$ is *a priori infeasible*. A priori infeasibility could be defined similarly for capacity constraints. However, assuming that the load associated to each customer is at most $C/2$, where C is the capacity of the vehicle, each transition is a priori feasible with respect to capacity. In other words, two customers i and j with loads satisfying $q_i \leqslant C/2$ and $q_j \leqslant C/2$ may be picked up and dropped of in any order without the capacity constraint being violated.

Note that if for any two nodes i and j, both transitions $i \to j$ and $j \to i$ are infeasible a priori, the entire problem is infeasible. Otherwise, either $i \to j$ or $j \to i$ or both are a priori feasible and the pickup and delivery nodes of customers can be divided among m preceding clusters by using the following rule.

Definition 1. Cluster C_k precedes cluster C_l if and only if the transition $x \to y$ is a priori infeasible for all $x \in C_l$ and $y \in C_k$. In this case, we shall use the notation $C_k \prec C_l$.

Clearly, assuming that there exists a feasible solution, the above precedence relation of clusters is a strict order: (i) $C \not\prec C$ for all clusters, (ii) $C_a \prec C_b$ implies $C_b \not\prec C_a$ and (iii) $C_b \prec C_c$ and $C_a \prec C_b$ implies $C_a \prec C_c$. In addition, note that if a single node i has an infinite time window $[-\infty,+\infty]$, there is only one cluster since all transitions $j \to i$ and $i \to j$ are feasible a priori.

In practice, the clusters can be determined by means of an adjacency matrix F for which $F_{ij} = 1$ if $i \to j$ is feasible and $F_{ij} = 0$ if $i \to j$ is infeasible a priori. By arranging the rows and columns suitably we get a block upper triangular matrix where each block corresponds to a cluster. Fig. 1 shows the adjacency matrix of a sample problem involving four customers. The rows and columns are arranged in a descending order with respect to row sums. From the matrix five clusters can be identified (blocks on the diagonal), namely $\{1^\uparrow\} \prec \{1^\downarrow,3^\uparrow,2^\uparrow\} \prec \{2^\downarrow\} \prec \{3^\downarrow,4^\uparrow\} \prec \{4^\downarrow\}$. Thus, since the positions of nodes $1^\uparrow$, $2^\downarrow$ and $4^\downarrow$ are fixed, the problem falls down to determining the optimal ordering of nodes $1^\downarrow$, $3^\uparrow$, $2^\uparrow$ and $3^\downarrow$, $4^\uparrow$. In general, by clustering the nodes a priori, a significant amount of insertions need not be checked for feasibility.

	$1^\uparrow$	$1^\downarrow$	$3^\uparrow$	$2^\uparrow$	$2^\downarrow$	$3^\downarrow$	$4^\uparrow$	$4^\downarrow$
$1^\uparrow$	1	1	1	1	1	1	1	1
$1^\downarrow$	0	1	1	1	1	1	1	1
$3^\uparrow$	0	1	1	1	1	1	1	1
$2^\uparrow$	0	1	1	1	1	1	1	1
$2^\downarrow$	0	0	0	0	1	1	1	1
$3^\downarrow$	0	0	0	0	0	1	1	1
$4^\uparrow$	0	0	0	0	0	1	1	1
$4^\downarrow$	0	0	0	0	0	0	0	1

Fig. 1. A priori adjacency matrix of a sample problem involving 4 customers. From the matrix five clusters can be identified, namely $\{1^\uparrow\} \prec \{1^\downarrow,3^\uparrow,2^\uparrow\} \prec \{2^\downarrow\} \prec \{3^\downarrow,4^\uparrow\} \prec \{4^\downarrow\}$.

2.3. Structure and complexity

The structure of the algorithm for the static case is as follows:

1. The set of feasible service sequences S_N with respect to all customers is determined.
2. The optimization part consists of the evaluation of the objective function for all potential sequences $s \in S_N$.

The number of insertions that are tested for feasibility for customer i is bounded above by the formula

$$n(i) \leqslant m_{i-1} \sum_{x=1}^{2i-1} x = \frac{m_{i-1}(2i)(2i-1)}{2}, \tag{7}$$

where m_{i-1} is the number of feasible service sequences with respect to customers $1,\ldots,i-1$. The inequality can be justified by the fact that not all possible insertions $I(s,i,h,k)$ have to be checked for feasibility. This is due to the fact that if a pickup point index h' proves to produce only infeasible solutions, there is no point in checking the feasibility of any of the combinations (h',k).

In the worst case, where capacity and time constraints are *not restrictive*, we get

$$m_1 = 1, \quad m_2 = (4\cdot 3)/2 = 6, \quad m_3 = 6(6\cdot 5)/2 = 90,$$

$$m_i = \frac{(2i)(2i-1)}{2}\,\frac{(2(i-1))(2(i-2)-1)}{2}\cdots\frac{4\cdot 3}{2} = \frac{(2i)!}{2^i}.$$

Thus the maximum number of insertions required in the solution of the static case is $\sum_{i=1}^{N} \frac{(2i)!}{2^i} \approx \sum_{i=1}^{N} 2^{1-i} i^{2i+\frac{1}{2}} \sqrt{\pi}$. Thus, in the worst case, the number of feasible solutions is of order $\mathcal{O}(\sqrt{N}(N^2/2)^N)$ and thus the computational complexity of the optimization part is also high, as all feasible routes have to be evaluated.

Compared to the computational complexity $\mathcal{O}(N^2 3^N)$ of the solution to the static DARP without time windows studied in Psaraftis (1980), these figures do not seem interesting. However, in this work it is assumed that the time constraints are relatively strict implying few feasible service sequences for each customer and thus the computational efficiency of the insertion algorithm is increased.

Suppose that, due to strict time (and capacity) constraints, the number of potential service sequences m_i for customers $1,\ldots,i$ is bounded by some function $m : \mathbb{N} \to \mathbb{N}$ such that $m(i) \leqslant \frac{(2i)!}{2^i}$. Then the number of insertions for customer i is bounded by $n(i) \leqslant \frac{m(i-1)(2i)(2i-1)}{2}$. The total number of insertions is bounded by $\sum_{i=1}^{N} \frac{m(i-1)(2i)^2(2i-1)}{2}$. For example, if $m(i) = kN$ for some constant k, the computational complexity of the screening phase is reduced to $\mathcal{O}(N^4)$. If $m(i) = k$, the computational complexity is reduced to $\mathcal{O}(N^3)$.

On the grounds of previous calculations, it can be stated that the advanced insertion algorithm will generally not be able to produce exact solutions efficiently in cases where the capacity and time constraints are not restrictive. However, if the number of feasible service sequences is bounded due to strict constraints, the algorithm will lead to an exact solution computationally inexpensively. It is evident that the computational effort of existing routing algorithms also decrease when the problem becomes highly restricted. However, no exact algorithms for the time constrained single-vehicle DARP with the generalized objective function (1) have previously been reported in the literature. Thus, the complexity can not be rightfully compared to existing exact algorithms for the single-vehicle dial-a-ride problem with time windows, since they are designed to solve a special case of the problem in which only the route duration is minimized.

In addition, the advanced insertion algorithm has a special property of being extendable to an adjustable heuristic, as described in the following subsection.

2.4. A heuristic extension

Even if the capacity and time constraints were not highly restrictive, the algorithm can be modified easily by bounding the size of the set S_i of service sequences, in which new customers are inserted, by including only a maximum of L service sequences for each customer i. More precisely, if after inserting customer i, the number of feasible service sequences with respect to customers $1, \ldots, i$ is larger than L, the set of feasible service sequences S_i with respect to customers $1, \ldots, i$ is narrowed by including only L service sequences, that seem to allow the insertion of remaining customers (see part Section 2.4.1). After the last customer has been inserted, the feasible service sequences are evaluated by means of the objective function (1).

This modification leads to a heuristic algorithm, in which the computational effort can be controlled by the parameter L, referred to as the *degree* of the heuristic. The resulting algorithm is somewhat sophisticated in a way that it produces globally optimal solutions for small sets of customers and when the number of customers is increased, the algorithm still produces locally optimal solutions with reasonable computational effort. In the special case where $L = 1$, the algorithm reduces to the classical insertion algorithm. If $L \geqslant \frac{(2N)!}{2^N}$, the heuristic coincides with the exact version of the algorithm as no routes are discarded.

2.4.1. Objective functions

In order to be able to efficiently make use of the above heuristic extension idea, the set of service sequences is narrowed by means of a certain heuristic objective function after the insertion of each customer. Since the main purpose of heuristics at the operational level is to always produce some implementable solutions very quickly, even if they were only locally optimal, such an objective function should be defined in a way that the algorithm is capable of producing *feasible* solutions even if the complexity of the problem was high.

Looking only at the cost defined in formula (1) may eliminate from consideration sequences that are marginally costlier but would easily allow the insertion of remaining customers in the route. Thus, more sophisticated criteria should be considered to help ensure that the heuristic will find a feasible solution when one exists.

In other words, the function should favor service sequences with enough *time slack* for those customers, that have not been inserted into the sequences. In this work, the following heuristic objectives are considered. Given the service sequence $s = (p_1, \ldots, p_m)$, we wish to optimize one of the functions

$$f_{rl}(s) = t_m, \quad \text{(Route duration)} \tag{8}$$

$$f_{ts}(s) = \sum_{j=1}^{m} l_j - t_j, \quad \text{(Total time slack)} \tag{9}$$

$$f_{min}(s) = \min_{j \in \{1,\ldots,m\}} l_j - t_j, \quad \text{(Max—min time slack)} \tag{10}$$

where $[e_j, l_j]$ is the time window and t_j is the calculated time of arrival at node p_j.

In general, each of the above objective functions aim to maximize the temporal flexibility of service sequences in different ways.

Route duration (8) favors service sequences in which the time to serve all customers is as small as possible. This objective can be justified by the fact that it is likely that new customers may be inserted at the rear of a route that is executed quickly.

Total time slack (9) stores sequences in which the sum of excess times (or the average excess time) at the nodes is maximized, that is, sequences which are likely to allow the insertion of a new customer before the last node.

Max–min (10) seeks sequences in which the minimum excess time at the nodes of the route is maximized. In other words, the sequences in which there is at least some time slack at each node are considered potential.

A simple example motivating the use of the above objective functions is presented in Fig. 2. In Section 4, the different objectives are compared by means of computational experiments.

2.4.2. Tuning

In order to ensure that the heuristic always produces a feasible solution when one exists, the parameter L can be tuned during runtime by using the following idea.

1. At first, the problem is solved by using an initial degree L_0.
2. Each time the algorithm is unable to find a feasible solution, the degree is increased and the problem is solved again.

At this point one might ask which initial value for L should be chosen and which tuning strategy would produce a feasible solution with least computational effort? These questions can be answered by studying the expected CPU time of the algorithm by means of the following model.

Let $P(S|L = l)$ denote the probability that a solution is found by using the degree $L = l$ and let $T(l)$ denote the average running time of the algorithm with $L = l$. The average CPU times of feasible runs are assumed equal to those occurring during infeasible runs, although infeasible runs are actually slightly less expensive. However, it can be seen that this approximation will not affect the results of the following examination. The expected CPU time for a given tuning strategy $(L_0, L_1, \ldots)$, where $L_0 < L_1 < L_2 < \ldots$, is given by the formula

$$T(L_0) + \sum_{i=1}^{\infty} T(L_i) \prod_{j=0}^{i-1} (1 - P(S|L = L_i)).$$

It should be noted that the above probabilities are actually conditional: if a solution is not found with some value of L, there is no point in solving the problem again with the same value. In addition, a relatively small increase in the value of L will not significantly affect the possibilities of finding a solution. Thus, the tuning approach used in the following examination is based on multiplying the value

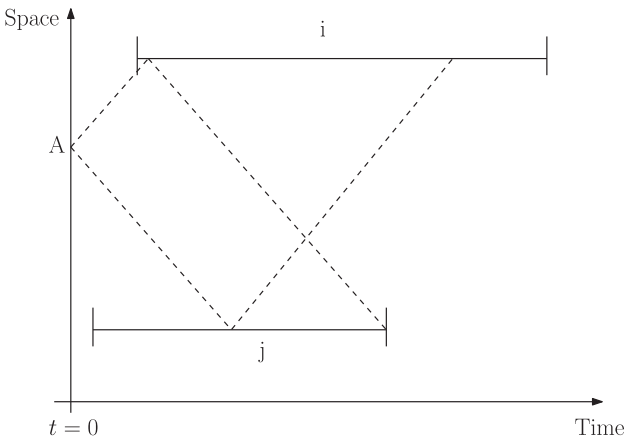


Fig. 2. Route flexibility. A vehicle is located at A at $t = 0$, and two customers are due to be picked up within the presented time windows at i and j. The dashed lines represent two possible routes for the vehicle. If the route duration were minimized, i should be visited before j. However, since there would be no "slack time" at j, this decision would a priori exclude the possibility that new customers could be inserted between i and j. On the other hand, if j were visited before i, there would be more possibilities for inserting new customers on the route before i. However, the route $A \to i \to j$ is shorter and thus it is more likely that customers can be inserted at the end of the sequence.

of L by a number K each time a feasible solution is not found. This method will be referred to as K-multiplying.

Assuming that the complexity grows linearly with l, that is $T(l) = l\,T(1)$, and that the probability of finding a solution is sufficiently high, it can be shown that the optimal initial degree for the K-multiplying method has to be relatively small. At first we will show that if $P(S|L \geqslant 1) > 1/2$, then any initial degree $L_0 = mK$, where $m \in \mathbb{N}$, is suboptimal.

Theorem 1. *Let $E(\tau(L_0))$ denote the expected CPU time of the K-multiplying method for the initial value L_0. If $T(l) = lT(1)$ for $l \in \mathbb{N}$ and $P(S|L \geqslant 1) > 1/2$, then*

$$E(\tau(L_0)) \leqslant E(\tau(KL_0))$$

for all $L_0 \in \mathbb{N}$ and $K \geqslant 2$.

Proof. Since it is clear that $E(\tau(l)) \geqslant T(l)$ for $l \in \mathbb{N}$, for any $K \geqslant 2$, we get

$$\begin{aligned} E(\tau(L_0)) &= T(L_0) + (1 - P(S|L = L_0))E(\tau(KL_0)) < T(L_0) + \frac{1}{2}E(\tau(KL_0)) \\ &= \frac{1}{k}T(KL_0) + \frac{1}{2}E(\tau(KL_0)) \leqslant \left(\frac{1}{K} + \frac{1}{2}\right)E(\tau(KL_0)) \\ &\leqslant E(\tau(KL_0)). \qquad \square \end{aligned}$$

Then, the following theorem states that if the probability of finding a solution is at least $1 - 1/K$, the optimal initial value is bounded above by the logarithm of the smallest natural number l for which $P(S|L = l) = 1$.

Theorem 2. *Let l' denote the smallest natural number l for which $P(S|L = l) = 1$. If $P(S|L \geqslant 1) > 1 - 1/K$, then*

$$E(\tau(L_0)) < T(L_0(1 + \log_K l'/L_0))$$

for all $L_0 \in \mathbb{N}$ and for all $K > 1$.

Proof. The expected CPU time is given by the formula

$$E(\tau(L_0)) = \sum_{i=0}^{\infty}(1 - P(S|L = K^i L_0))^i T(K^i L_0).$$

Since $(1 - P(S|L = K^i L_0)) = 0$ for $K^i L_0 > l'$, that is, $i > \log_K l'/L_0$, we get

$$\begin{aligned} E(\tau(L_0)) &= \sum_{i=0}^{\lfloor \log_K l'/L_0 \rfloor}(1 - P(S|L = K^i L_0))^i T(K^i L_0) \\ &\leqslant \sum_{i=0}^{\lfloor \log_K l'/L_0 \rfloor}(1 - (1 - 1/K))^i K^i T(L_0) \\ &\leqslant T(L_0)(1 + \log_K l'/L_0). \qquad \square \end{aligned}$$

In general, the two theorems imply that the expected CPU time is minimized when the initial degree L_0 is small. However, this is true only if the probability of finding a solution with small values of L is relatively high. For example, if $l' = 16$, $K = 2$ and $P(S|L \geqslant 1) > 0.5$, Theorem 2 states that $E(\tau(1)) < T(5)$. By Theorem 1 we know that the values 2 and 4 are suboptimal for $K = 2$. Thus, the optimal initial value is either 1 or 3 depending on the actual probabilities of finding solutions with different values of L. The fact that the optimal value of L_0 (with respect to complexity) is small is verified by the computational results presented in Section 4.

Finally, it should be noted that the optimal choice of K and L_0 will in general depend on the type of the problem to be solved and that by increasing the values, the optimality of the solution with respect to the objective of the problem is increased as well since more sequences are evaluated, as will be seen in Section 4.

3. The dynamic case

Let us discuss the extension of the predescribed algorithm to the dynamic case, in which new customers are appended to the route in real time.Generally, it can be seen that there is no need to compute a new route except when a new customer request occurs, a customer does not show up at the agreed pickup location, the vehicle is ahead or behind of schedule or the vehicle has lost its way.

A simple example, similar to the one presented in Psaraftis (1980), on the real-time updating of a vehicle route is shown in Fig. 3. Initially, customers 1 and 2 have been assigned pickup and delivery points and corresponding time windows. The vehicle located at A follows the tentative optimal route $(1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow})$, where "$\uparrow$"-symbols denote pickup nodes and "$\downarrow$"-symbols denote delivery nodes. (see Fig. 3a). At the time the vehicle is at B, the vehicle route is updated with respect to customers 1, 2 and 3. At this instant a new tentative optimal route shown in Fig. 3b is produced.

Note that the new route does not have B as origin, but a point B', slightly ahead of B. This is due to two facts: (i) It will take some time for the algorithm to process the new input and reoptimize, (ii) It will take some time for the driver of the vehicle to process the information regarding the new route. The distance between B and B' depends in general on the particular dial-a-ride service examined. For example, in a highly dynamic setting, in which the modification of the route is not allowed to take more than a few seconds, it may be assumed that the latter of the facts is more restrictive. Any dynamic dial-a-ride service should make use of a mechanism to update the point B' at sufficient time intervals. In this work, however, it is assumed that the point B' is known at each instant, as well as the estimated time of arrival T' at B'. The *vehicle checkpoint* (B', T') acts as the starting point for all service sequences.

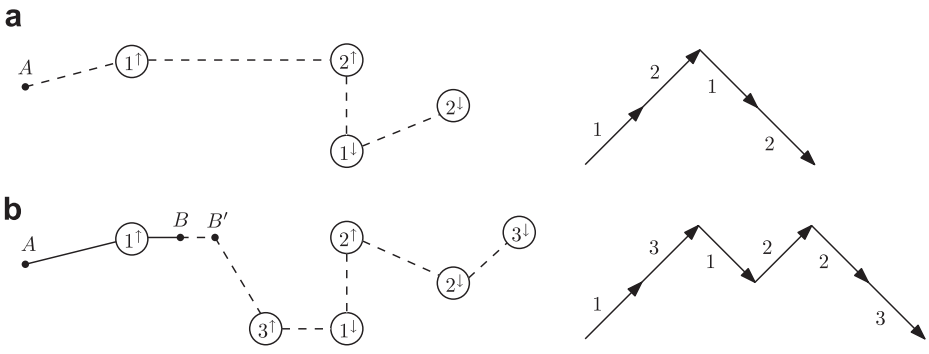


Fig. 3. Modifications in the vehicle route. The route is updated when the vehicle is at B and a new tentative optimal route beginning at B' is produced. The figures on the right show the routes as so-called labeled Dyck paths (Cori, 2009), in which each pickup $i^{\uparrow}$ precedes the corresponding drop-off $i^{\downarrow}$. At the time a new customer is added to the route, a new path is formed. Clearly, the "height" of the path shows the number of customers aboard in different parts of the route.

For customers that have already been picked up, the pickup point is not a part of the input of the problem. Thus, for such customers only the delivery point is considered in the insertion procedure. More precisely, given a service sequence $s = (p_1, \ldots, p_m)$, the insertion function $I'(s,i,k)$ is used for all $k \in \{1, \ldots, m+1\}$, where the insertion $I'(s,i,k)$ produces the service sequence $(p_1, \ldots, p_{k-1}, N+i, p_k, \ldots, p_m)$.

In dynamic models, the objective function used to find the solution over a rolling horizon has often little to do with the measures developed to evaluate the overall quality of a solution (Powell et al., 1995). However, the advanced insertion method is designed in a way that several objective functions, that are thought to be suitable for the dynamic problem, may be incorporated with minimal work. For example, if the vehicle route is subject to several modifications in short periods of time, one of the flexibility measures (10), (9) or forward time slack defined in Savelsbergh (1992) may be used as an objective function for the problem in order to be able to insert as many future customers in the route as possible. Generally, the choice of the objective function depends on the particular version of the dynamic routing problem and the performance of different objective functions may be sensitive to, for example, constraints, demand intensity or the number of vehicles. For a study related to choosing an objective function for the immediate-request DARP, the reader is referred to Hyytiä et al. (2010).

4. Numerical experiments

In the following paragraphs, a collection of computational results obtained by the advanced insertion method are proffered. The exact and heuristic versions of the algorithm were tested on a set of problems involving different numbers of customers and different time window widths determined by a travel time ratio R describing the maximum allowed ratio of travel time to direct ride time. The pickup and drop-off points of customers were chosen randomly from a square-shaped service area and the ride times between the points were modeled by euclidean distances.

At first, the complexity of the problem was studied with respect to three parameters, namely (i) the number of customers N, (ii) travel time ratio R and (iii) the average time interval μ between customer requests. The complexity of the exact algorithm was measured in terms of the average number of feasible sequences in *feasible problem instances*, that is, randomized problems for which at least one feasible solution was found. The number of feasible sequences refers to the number of feasible sequences after inserting the last customer, even though in some cases the number of feasible sequences for customers before the last one may be greater. However, using the final number of feasible sequences as a measure of complexity can be justified by the fact that the number of feasible sequences was seen to be an increasing function of the number of inserted customers in most studied cases (67% of feasible instances with $N = 20$, $R = 3$ and $\mu = 1800$). In addition, the measure is general in a sense that it is independent of the insertion order of customers. In fact, the final number of feasible sequences gives us an insight on the complexity of the problem itself, since any enumeration algorithm would have to evaluate the same number of feasible sequences.

Then, the performance of the heuristic with different objective functions and different degrees was evaluated. The complexity was measured in terms of the number of sequences evaluated by the heuristic.

A summary of performed experiments and the corresponding results is shown in Table 2.

4.1. Parameters

The parameters of the experiments are based on a demand-responsive transport service operating in a neighborhood of 100 km^2. For simplicity, a square-shaped service area is studied. The largest travel time ratio used in the experiments is 3, which describes a relatively low level of service. The number of customers assigned to a single vehicle at a certain instant is strongly dependent on the type of the dial-a-ride service. In highly dynamic services, in which most trips are requested not long before the trip is due to begin, the number of customers in the tentative route is relatively small (see Hyytiä et al., 2010). On the other hand, if trips are requested hours in advance, the planning horizon is significantly longer and thus the complexity of the problem is increased. In the experiments, no pre-order time limits are assumed. The demand is generated by means of a Poisson process, in a way that the single vehicle serves up to 10 customers per hour on average, which is considered a relatively high value for vehicle efficiency. The vehicles used in demand-responsive transport services are often small or medium sized and thus the number of customers assigned to a single-vehicle route is often limited to 10–20 customers. In the experiments, the vehicle capacity is set to $C = 10$ and the focus is on problem instances with up to 20 customers. In experiment 6, we study the robustness of the algorithm in unexpected situations, in which there may be up to 50 customers assigned to the vehicle.

4.2. Data generation

Letting N denote the number of customers, R denote the travel time ratio and μ denote the average time interval between customers, the data used in the following experiments was generated as follows. For each customer $i \in \{1, \ldots, N\}$:

1. Choose a pickup point u_i and a drop-off point u_{N+i} randomly in $[0, 10, 000] \times [0, 10, 000] \subset \mathbb{R}^2$ (unit = 1 meter).
2. Determine the earliest pickup time e_i by means of a Poisson process with intensity $\lambda = 1/\mu$.

Table 2
A summary of performed experiments.

Experiment number	Algorithm	Travel time ratio	Number of customers	Goal	Main result	Figure number
1	Exact	2, 2.5, 3	1 …, 20	Complexity with respect to number of customers	$\mathcal{O}(\exp(N))$	Fig. 4
2	Exact	1.5, …, 3	5, 10, 20	Complexity with respect to travel time index	$> \mathcal{O}(\exp(R))$	Fig. 5
3	Exact	3	10	Complexity with respect to time interval	$\mathcal{O}(\exp(-\mu))$	Fig. 6
4	Heuristic	3	20	Compare heuristic cost functions	Total time slack	Fig. 7
5	Tuned heuristic	3	20	Complexity and error with respect to L_0	Error decreases when L_0 is increased	Fig. 8
6	Tuned heuristic	3	2, …, 50	Robustness with respect to number of customers	Feasible solutions with little effort	Fig. 9

3. Determine the latest drop-off time l_{N+1} by means of the formula $l_{N+i} = e_i + Rt(u_i, u_{N+i})$, where $t(u_i, u_{N+i}) = \|u_i - u_{N+1}\|/40$ kilometers/hour is the direct ride time from the pickup point to the drop-off point.
4. Determine the latest pickup time by $l_i = l_{N+i} - t(u_i, u_{N+i})$ and the earliest drop-off time by $e_{N+i} = e_i + t(u_i, u_{N+i})$.

For simplicity, both pickup and drop-off points were considered for each customer in all experiments, even though the pickup points of customers on board need not be considered in the dynamic case. However, this property of the dynamic case indicates that enumerating all feasible sequences during the execution of a route requires less computational work than complete enumeration in the corresponding static case, in which pickup points of all customers are considered as well. Thus, the results to be presented act as an upper bound for the dynamic case.

4.3. Experiments

The following experiments were performed on a standard laptop computer with a 2.2 GHz processor. The CPU times and the number of evaluated sequences appeared to have a roughly linear relationship. A typical problem instance involving 20 customers could be solved up to optimality within less than a second.

4.3.1. Experiment 1: Number of customers

At first we study the complexity of the exact algorithm with respect to the number of customers. In practice, the number of customers assigned to a single vehicle is governed by the pre-order time of the dial-a-ride service: if customers may request service in advance, the planned vehicle routes are expected to be longer than in immediate-request services. Fig. 4 shows the average number of feasible sequences in feasible problem instances on a logarithmic scale, computed over 10,000 randomized instances for $N = 1, \ldots, 20$, $R = 2$, 2.5, 3 and $\mu = 1800$ seconds.

Referring to the figure, it can be seen that the complexity of the problem increases exponentially with respect to the number of customers with all studied values of the travel time ratio. In addition, the complexity is increased with the travel time ratio.

4.3.2. Experiment 2: Travel time ratio

Let us study the complexity of the exact algorithm as a function of travel time ratio. Fig. 5 shows the average number of feasible sequences in feasible problem instances on a logarithmic scale, computed over 10,000 randomized instances for $R = 1.5, \ldots, 3$, $N = 5$, 10, 20 and $\mu = 1800$ seconds.

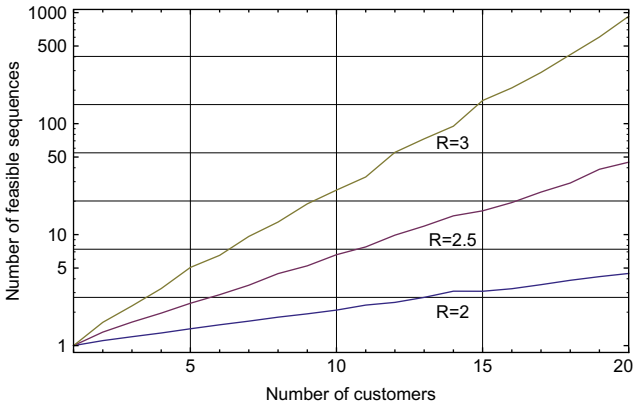


Fig. 4. Experiment 1. The complexity of the exact algorithm with respect to the number of customers on a logarithmic scale. The three curves represent, as a function of the number of customers, the average number of feasible sequences for $R = 2$, 2.5, 3 and $\mu = 1800$ seconds. Clearly, the complexity increases exponentially with respect to the number of customers.

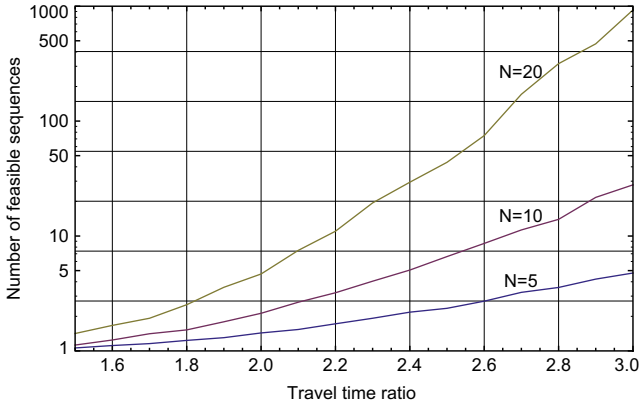


Fig. 5. Experiment 2. The complexity of the exact algorithm with respect to travel time ratio on a logarithmic scale. The three curves represent, as a function of the travel time ratio, the average number of feasible sequences for $N = 5$, 10, 20 and $\mu = 1800$ seconds.

The figure shows that the effect of the travel time ratio on the complexity of the problem is significant. The fact that the slopes of the curves increase with R on the logarithmic scale indicates that the relation between complexity and R is superexponential.

4.3.3. Experiment 3: Time interval

Let us conclude the study of the exact algorithm by examining complexity with respect to the average time interval between customer requests. The solid lines in Fig. 6 represent the average number of feasible sequences in (i) feasible problem instances and (ii) all problem instances, on a logarithmic scale, computed over 10,000 randomized instances for $N = 10$, $R = 3$ and $\mu = 6, \ldots, 40$ minutes. The dashed line corresponds to the fraction of problem instances, for which at least one feasible solution was found.

The figure indicates that the complexity of feasible problem instances decreases exponentially with respect to the average time interval μ. On the other hand, the probability of finding at least one feasible solution is increased with μ. By looking at the curve corresponding to the average complexity of all problem instances (including infeasible cases), it can be seen that the complexity is maximized at a certain time interval ($\mu = 24$ minutes in this case), in which both the probability of finding a feasible solution and the number of feasible sequences in feasible cases are relatively large.

At this point, it should be emphasized that the above results apply to randomized instances for the single-vehicle problem. In a dynamic multiple-vehicle setting, an effort is made to divide the customers among available vehicles optimally. Thus, it is suggested that a larger number of customers can be served by a single vehicle in less time than in the case in which the customers' pickup and drop-off locations are completely random. However, the above results give us an insight on how the complexity of the single-vehicle subroutine behaves with respect to the average time interval between the earliest pickup times of customers assigned to a single vehicle.

4.3.4. Experiment 4: Objective functions

Let us study the performance of the three different heuristic cost functions (8)–(10) as a function of the degree L of the heuristic. Fig. 7 shows the fraction of problem instances for which a feasible solution was found by the heuristic (compared to the exact algorithm), computed over 10,000 randomized instances. The number of customers was set to $N = 20$, the travel time ratio was 3 and the average time interval $\mu = 1800$ seconds was used.

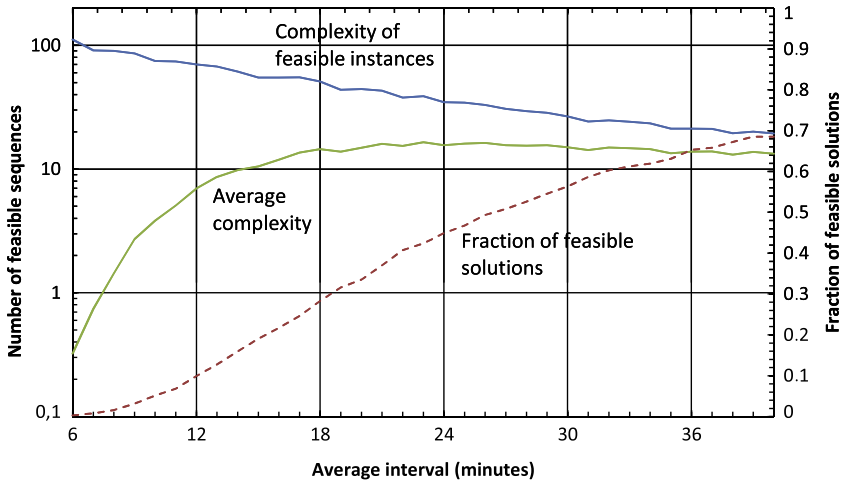


Fig. 6. Experiment 3. The complexity of the exact algorithm with respect to the average time interval between customers. The solid lines represent the average number of feasible sequences in feasible problem instances and all problem instances, on a logarithmic scale for $N = 10$ and $R = 3$. The dashed line corresponds to the fraction of problem instances, for which at least one feasible solution was found.

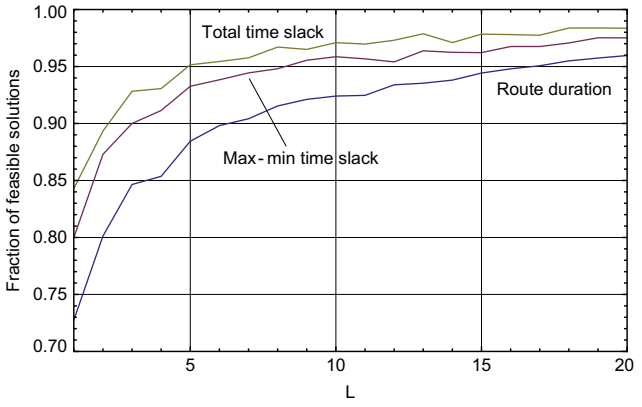
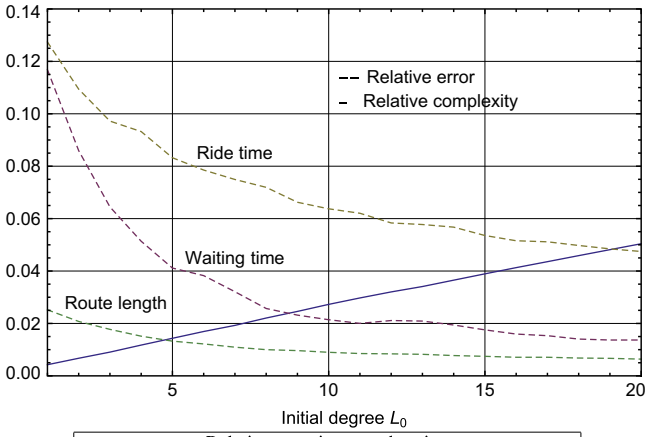


Fig. 7. Experiment 4. The performance of three different heuristic objective functions as functions of degree L. The curves represent the fractions of instances for which a feasible solution was found by the objective functions (compared to the exact algorithm), for $N = 20$, $R = 3$ and $\mu = 1800$ seconds. The total slack time objective function outperforms the other two in all studied cases.



Relative error in route duration							
L_0	1	2	3	4	5	6	7
Relative error (10^{-5})	9.8	4.9	2.4	2.4	2.4	2.4	0

Fig. 8. Experiment 5. Relative complexity and relative error of the tuned heuristic as a function of the initial degree L_0. The solid line shows the relative complexity of the tuned heuristic, compared to the exact algorithm and the dashed lines correspond to the relative error in total ride time and total waiting time of customers and in route length, compared to the exact algorithm for $N = 20$ and $R = 3$. The table below the figure shows the corresponding relative error in route duration. Clearly, the heuristic is capable of finding short routes with little effort. As for more complex objectives, such as total ride time and total waiting time, more work is needed to control the error.

Referring to the figure, it can be seen that the total time slack cost function (9) is capable of finding a feasible solution to randomized problems most often, while the performance of the route duration cost function (8) is worst of the three algorithms. Note that as the degree L is increased, the fraction of feasible solutions converges to 1 for any heuristic cost function, since whenever $L \geqslant \frac{(2N)!}{2^N}$, the heuristic coincides with the exact algorithm regardless of the studied problem.

4.3.5. Experiment 5: Relative error

Let us study the complexity and relative error of the tuned heuristic with respect to the initial degree L_0. The solid line in Fig. 8 shows the relative complexity of the tuned heuristic, compared to the exact algorithm and the dashed lines correspond to the relative error in total ride time, total waiting time and route length, compared to the exact algorithm, calculated over 10,000 randomized runs. The table below the figure shows the corresponding relative error in route duration.

The relative complexity values were computed by means of the formula $\sum_{k\in F} h_k / \sum_{k\in F} H_k$, where h_k and H_k denote the number of sequences evaluated by the tuned heuristic and the exact algorithm in feasible problem instance k and F denotes the set of feasible problem instances. The relative error values for total ride time were computed by means of the formula $\sum_{k\in F} r_k / \sum_{k\in F} R_k - 1$, where r_k and R_k denote the sum of ride times $\left(\sum_{i=1}^{N} T_i^R\right)$ of N customers in feasible problem instance k. The relative error for total waiting time, route length and route duration were calculated in a similar fashion.

As in the previous experiment, the number of customers was set to $N = 20$, the travel time ratio was 3 and the average time interval $\mu = 1800$ seconds was used. The multiplier K of the tuned heuristic was set equal to 2, that is, each time a feasible solution was not found, the degree of the heuristic was doubled and the problem was solved again.

By looking at the figure, it can be seen that the complexity of the tuned heuristic is a linear function of the initial degree L_0. On the

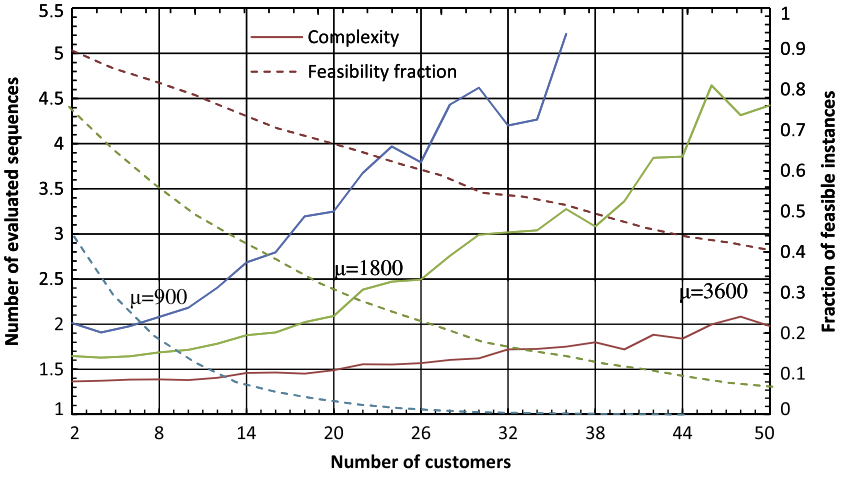


Fig. 9. Experiment 6. Robustness of the tuned heuristic as a function of the number of customers. The solid lines show the complexity of the tuned heuristic in terms of the average number of evaluated sequences in feasible instances and the dashed lines correspond to the fraction of instances with at least one feasible solution for $\mu \in \{900, 1800, 3600\}$, $R = 3$ and $L_0 = 1$. The heuristic is capable of finding feasible routes for a large number of customers with little effort. The complexity of feasible instances is increased as the time interval between customers is decreased, but the fraction of feasible problems is decreased with μ.

other hand, the error in both total ride time and total waiting time is decreased when L_0 is increased. The error in total waiting time is relatively smaller than the corresponding error in total ride time. By looking at the curve corresponding to route length and the table below the figure it can be seen that the heuristic is capable of finding short routes close to the optimal solution with little effort. As for more complex objectives, such as total ride time and total waiting time, more work is needed to find good quality sequences. Thus, it can be suggested that using a large initial degree L_0 is well motivated only with complex objective functions.

4.3.6. Experiment 6: Robustness

Finally, we examine the robustness of the tuned heuristic as a function of the number of customers. The solid lines in Fig. 9 show the complexity of the tuned heuristic in terms of the average number of evaluated sequences in feasible instances and the dashed lines correspond to the fraction of problem instances with at least one feasible solution, computed over 10,000 randomized runs for $\mu \in \{900, 1800, 3600\}$, $R = 3$ and $L_0 = 1$. As in the previous experiment, the multiplier $K = 2$ was used.

Clearly, the heuristic is capable of finding feasible routes for a large number of customers with little effort. While the complexity of feasible instances is increased as the average time interval between customers is decreased, the fraction of problems with at least one feasible solution decreases with μ. By looking at the dashed curves, it can be seen that the probability of finding a feasible solution tends to zero as the number of customers is increased. The above observations imply that with randomized data, problems for which the solution is difficult to find are relatively uncommon.

As a conclusion, a significant advantage of using the heuristic algorithm can be identified as the possibility of being able to control the computational effort smoothly. While the case $L = 1$ coincides with the classical insertion algorithm in which the complexity and the probability of finding a feasible solution is relatively low, by increasing the value of L, the CPU time, as well as the overall quality of the solution, is increased. In addition, the heuristic may also be used to always produce a solution whenever one exists by tuning it during run-time (see K-multiplying method, part Section 2.4.2).

5. Conclusions

In this work, an exact optimization procedure is developed to solve the static and dynamic versions of the single-vehicle dial-a-ride problem with time windows. Using complete enumeration to solve the problem with respect to a generalized objective function is motivated by the dynamic nature of online demand-responsive transport services, in which looking only at the tentative route duration, as in existing algorithms for the problem, may decrease the possibilities of serving future customers. In addition, an adjustable heuristic extension to the algorithm is introduced, in order to be able to control the CPU times: if the problem size is reasonable, the proposed solution method produces globally optimal solutions. If the problem size is increased, the algorithm adjusts itself to produce locally optimal solutions, closing the gap between the classical insertion heuristic and the exact solution and thus making the algorithm applicable to any static or dynamic dial-a-ride problem.

Acknowledgments

This work was partly supported by the Finnish Funding Agency for Technology and Innovation, Finnish Ministry of Transport and Communications, Helsinki Metropolitan Area Council and Helsinki City Transport. The author is most indebted to Dr. Harri Hakula for his assistance and would also like to thank Dr. Aleksi Penttinen, Dr. Esa Hyytiä and three anonymous referees for their insightful comments to improve the quality of the paper.

References

Attanasio, A., Cordeau, J.-F., Ghiani, G., Laporte, G., 2004. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Computing 30, 377–387.

Bent, R., Van Hentenryck, P., 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research 33 (4), 875–893.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. European Journal of Operational Research 202, 8–15.

Bianco, L., Mingozzi, A., Ricciardelli, S., Spadoni, M., 1994. Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. INFOR 32, 19–31.

Cordeau, J.-F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Operations Research 54, 573–586.

Cordeau, J.-F., Laporte, G., 2003a. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research B 37, 579–594.

Cordeau, J.-F., Laporte, G., 2003b. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. 4OR: A Quarterly Journal of Operations Research 1, 89–101.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: Models and algorithms. Annals of Operations Research 153, 29–46.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., Savelsbergh, M., 2007. Transportation on demand. In: Transportation. North-Holland, Amsterdam, pp. 429–466.

Cori, R., 2009. Indecomposable permutations, hypermaps and labeled dyck paths. Journal of Combinatorial Theory, Series A 116 (8), 1326–1343.
Coslovich, L., Pesenti, R., Ukovich, W., 2006. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. European Journal of Operational Research 175, 1605–1615.
Desrosiers, J., Dumas, Y., Soumis, F., 1986. A dynamic programming solution of the large-scale single vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences 6, 301–325.
Diana, M., Dessouky, M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transportation Research Part B 38, 539–557.
Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7–22.
Garaix, T., Artigues, C., Feillet, D., Josselin, D., 2010. Vehicle routing problems with alternative paths: An application to on-demand transportation. European Journal of Operational Research 204, 62–75.
Hernández-Pérez, H., Salazar-González, J.-J., 2009. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. European Journal of Operational Research 196, 987–995.
Horn, M., 2002. Fleet scheduling and dispatching for demand-responsive passenger services. Transportation Research Part C 10, 35–63.
Hunsaker, B., Savelsbergh, M.W.P., 2002. Efficient feasibility testing for dial-a-ride problems. Operations Research Letters 30, 169–173.
Hyytiä, E., Häme, L., Penttinen, A., Sulonen, R., 2010. Simulation of a large scale dynamic pickup and delivery problem. In: Third International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010).
Jaw, J., Odoni, A., Psaraftis, H., Wilson, N., 1986. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation Research Part B 20, 243–257.
Madsen, O., Ravn, H., Rygaard, J., 1995. A heuristic algorithm for the a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Annals of Operations Research 60, 193–208.
Melachrinoudis, E., Ilhan, A.B., Min, H., 2007. A dial-a-ride problem for client transportation in a healthcare organization. Computers & Operations Research 34, 742–759.
Papadimitriou, C., 1977. The euclidean travelling salesman problem is np-complete. Theoret. Comput. Sci. 4, 237–244.
Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 37, 1129–1138.
Powell, W.B., Jaillet, P., Odoni, A., 1995. Stochastic and dynamic networks and routing. Network Routing, vol. 8. North-Holland, Amsterdam, pp. 141–295.
Psaraftis, H., 1980. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. Transportation Science 14, 130–154.
Psaraftis, H., 1983. An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. Transportation Science 17, 351–357.
Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40 (4), 455–472.
Savelsbergh, M., 1992. The vehicle routing problem with time windows: Minimizing route duration. ORSA Journal on Computing 4, 146–154.
Sexton, T., 1979. The single vehicle many-to-many routing and scheduling problem. Ph.D. Dissertation, SUNY at Stony Brook.
Sexton, T., Bodin, L.D., 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. Transportation Science 19, 378–410.
Sexton, T., Bodin, L.D., 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: Ii routing. Transportation Science 19, 411–435.
Toth, P., Vigo, D., 1997. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science 31, 60–71.
Wong, K., Bell, M., 2006. Solution of the dial-a-ride problem with multi-dimensional capacity constraints. International Transactions in Operational Research 13, 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. European Journal of Operational Research 174, 1117–1139.
Xiang, Z., Chu, C., Chen, H., 2008. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. European Journal of Operational Research 185, 534–551.