Math Meth Oper Res (2009) 69:323–341
DOI 10.1007/s00186-008-0262-5

ORIGINAL ARTICLE

A two-stage model for a day-ahead paratransit planning problem

Maria L. A. G. Cremers · Willem K. Klein Haneveld · Maarten H. van der Vlerk

Received: 28 September 2006 / Accepted: 17 December 2007 / Published online: 20 September 2008

Abstract We consider a dynamic planning problem for paratransit transportation. The focus is on a decision to take one day ahead: which requests to serve with own vehicles, and which requests to subcontract to taxis? We call this problem the day-ahead paratransit planning problem. The developed model is a non-standard two-stage integer recourse model. Both stages consist of two consecutive optimization problems: the clustering of requests into routes, and the assignment of these routes to vehicles. To solve this model, a genetic algorithm approach is used. Computational results are presented for randomly generated data sets.

Keywords Paratransit transport · Stochastic programming · Genetic algorithm

Mathematics Subject Classification (2000) 90C15 · 90C27 · 90C59

1 Introduction

We consider a dynamic planning problem for the transport of elderly and disabled people. In particular, we focus on a decision to take one day before the day of operation (i.e., day-ahead): which requests should be served with own vehicles, and which requests should be assigned to taxis (i.e., subcontracted)? We will call this problem the Day-ahead Paratransit Planning problem (the DaPP problem). A typical aspect of the DaPP problem in particular and transportation of elderly and disabled people in general is that part of the requests is known one day ahead, but also part of the requests becomes known only on the day of operation itself. We pay special attention

M. L. A. G. Cremers (✉) · W. K. Klein Haneveld · M. H. van der Vlerk
Department of Operations, University of Groningen,
P. O. Box 800, 9700 AV Groningen, The Netherlands
e-mail: m.l.a.g.cremers@rug.nl

to this aspect of uncertainty with respect to late requests. Besides uncertainty, clustering (combining requests) is an important aspect too. Clustering of requests ensures that less vehicles are needed to serve all requests. The criterion upon which the decision of the DaPP problem is taken, is the expected costs of serving all requests. These costs depend to a large extent on the degree of clustering.

Several studies exist which examine the door-to-door transportation problem of elderly and disabled people (e.g. Attanasio et al. 2004; Borndörfer et al. 1997; Cordeau and Laporte 2003a,b; Jørgensen et al. 2007; Madsen et al. 1995; Toth and Vigo 1997). In this section, we first give a description of this problem and then present an overview of studies in the literature. The focus of this overview is on how different studies handle the use of taxis in their model. Explicitly incorporating taxis as part of the vehicle fleet, and deciding (day-ahead) about requests either to assign them to an own vehicle or to a taxi, results in the development of the DaPP problem. To the best of our knowledge, the DaPP problem has not been studied before.

In the literature, the dial-a-ride problem (DARP) is frequently used for the door-to-door transportation services of elderly and disabled people (paratransit services). In DARP, a vehicle fleet is given, as well as a set of service requests. Each request consists of an origin (pickup location), destination (delivery location), and desired time windows for pickup and delivery. The set of requests can either be known beforehand (static) or gradually revealed during the day of operation (dynamic). The aim is to find a set of minimum cost vehicle routes subject to constraints regarding, among others, vehicle capacity and time windows.

In the context of paratransit services, time windows are usually modeled as soft constraints. That is, deviations from the desired pickup and delivery times are not forbidden but included in customer service measures, together with waiting time and ride time. These measures can either be included in the (multi-criteria) objective function or added to the set of constraints. In the last case, a bound for each measure has to be prescribed.

Furthermore, in paratransit transport there are various types of customers, mostly depending on physical characteristics (e.g., ambulatory customers and customers in folding or non-folding wheelchairs). Customers often have two requests on the same day: an outbound request from home to a destination (e.g., a hospital), and an inbound request for the return trip. Typically, in paratransit service also the vehicle fleet is heterogeneous. Each vehicle type has its own capacity depending on the customer type. For example, a van has capacity for two wheelchairs and five ambulatory customers, or for four wheelchairs and two ambulatory customers. Sometimes taxis are used in addition to the own vehicle fleet, mainly for the transport of ambulatory customers in peak hours. Without the use of taxis it may not be possible to serve all requests, and often maximization of the number of requests served is then part of the problem.

In most studies, the model can handle more than one option on how to deal with taxis. Which option will be chosen depends on the planning operator. The algorithm of Jaw et al. (1986) can either reject customers or not. If rejection of customers is not allowed an additional vehicle is introduced. This additional vehicle will be part of the vehicle fleet from that time on, contrary to taxis which are available to serve one single request. The underlying assumption is a (more or less) unbounded number of available vehicles. Madsen et al. (1995) distinguish two options for the use of taxis. In the first option, they add taxis to the vehicle fleet with a cost larger than for vehicles of the own fleet. In the second option, they allow rejection of customers and assign rejected customers to taxis afterwards. Toth and Vigo (1997) have only one option which is the use of taxis, but only if strictly necessary and for non 'difficult' requests, which are in their specification ambulatory customers, those in folding wheelchairs, and customers not requiring specialized accompanying personnel. To minimize the use of taxis they introduce penalty costs.

Of all studies mentioned above only the studies of Madsen et al. (1995) and Attanasio et al. (2004) consider the dynamic version of the paratransit problem: at the beginning of the day of operation many requests are already known, but during the day more requests are revealed. All other studies consider the static version of the problem in which all requests are known beforehand. It is obvious that the static version is easier to solve, but the dynamic version should give better results. When one decides to study the dynamic version of the problem, different options exist to model the uncertainty of the late requests. Both Madsen et al. (1995) and Attanasio et al. (2004) use an insertion heuristic to insert the new requests in the existing vehicle routes.

As opposed to all other studies, which look at the paratransit problem on the day of operation, we consider the paratransit problem one day before the day of operation. By then, many requests are already known which makes it possible to assign already then some of the known requests to taxis. This is exactly the decision taken in the DaPP problem: which requests to assign day-ahead, when only part of the requests is known, to subcontractors. The advantages of considering the paratransit problem one day ahead are threefold. First, better rates can be negotiated with taxi companies one day before the day of operation than on the day itself. Second, the set of requests requiring scheduling on the day of operation is smaller so that a solution can be found faster. Third, since a day-ahead decision is made, the vehicle schedules need not be determined in detail. Some extra, simplifying assumptions can be made which diminish the complexity of the problem and resulting model.

In the DaPP problem, we explicitly consider taxis as part of the vehicle fleet. Furthermore, besides the known requests we also take late requests into account which we assume to become known at the beginning of the day of operation. Consequently, two decisions need to be made since in between more information becomes available: day-ahead a decision is made on which of the early requests will be subcontracted, using only probabilistic information on the late requests. Then, at the start of the day of operation, the late requests become known, and additional decisions are made taking this new information into account. That is, it is decided which of the remaining early requests and the late requests will be subcontracted. Thus, we have chosen to take the late requests into account and assume probabilistic information about them. This makes the problem more difficult to solve, but the main decision improves, as we will show later on (see Sect. 5). Our simplifying assumption on the flow of information allows to use a two-stage recourse model. In practice, new requests become known gradually during the day of operation, which could be formulated as a multi-stage recourse model. This approach will be analyzed in a subsequent paper, see also Sect. 6.

In Sect. 2 we give an extensive problem description of the DaPP problem. In Sect. 3, we present a two-stage recourse model. This model is non-standard since each stage consists of two optimization problems which have to be solved consecutively. Sect. 4 contains the heuristics used to solve both optimization problems. Furthermore, we present the genetic algorithm used to solve the two-stage recourse model as a whole. A description of the numerical experiments and results can be found in Sect. 5, followed by conclusions and recommendations for further research in Sect. 6.

2 Problem description

In the DaPP problem it is determined which requests in paratransit service are best to assign to taxis one day before the day of operation, under uncertainty of the requests that become known on the day of operation itself. There are two sets of requests, a set of known or early requests, and a set of late requests which we assume to become known at the beginning of the day of operation. All requests need to be served. Each request, whether it is early or late, is characterized by the number of customers, an origin (or pickup location), and destination (also called delivery location). One time window is specified by the customer(s), either for pickup or for delivery. For outbound requests the time window for delivery is specified by the customer(s), and for inbound requests the time window for pickup is specified by the customer(s).

There are two types of customers: ambulatory and those in wheelchairs, and three types of vehicles: cars, vans, and taxis. Cars and vans belong to the own vehicle fleet while taxis do not. Cars can only be used for ambulatory customers, and vans and taxis have convertible seats such that both ambulatory and wheelchair customers can be transported. For vehicle types of the own fleet (cars and vans) the capacity is given, as well as the available number per time period. We assume the capacity of cars to be smaller than that of vans. For taxis, the capacity is not relevant in this problem and we assume that each time period an unlimited number is available. Without loss of generality, we may assume that the number of customers of a request is at most equal to the capacity of a van, such that all requests can be served by one vehicle. If not, the request can be divided into two or more requests.

The average driving speed is the same for all vehicle types. We assume that stopping to pick up or deliver customers does not take time. Furthermore, when a vehicle becomes empty after finishing a request, the vehicle becomes immediately available to serve a new request, i.e., driving empty from the destination of the current request to the origin of the new request does not take time. Both assumptions are simplifications of reality which are acceptable in our view, since we are concentrating on a day-ahead problem which does not require the actual vehicle schedule to be made.

We use the (expected) operational costs as decision criterion in the DaPP problem. For every vehicle type, the operational costs are proportional to the distance driven, and for taxis a fee is added for each request that is assigned to a taxi. Moreover, subcontracting on the day of operation is more expensive than day-ahead. The fixed costs of the own vehicles are not taken into account.

Clustering is an important aspect in the paratransit problem, and in some studies it even is the only subject under consideration (e.g. Desrosiers et al. 1991;

Ioachim et al. 1995). Clustering ensures that less vehicles are needed to serve all requests. As a result, serving clustered requests is cheaper than serving the requests separately. Also in the DaPP problem, clustering is an important aspect. Since less vehicles are needed to serve all requests and taxis are more expensive than own vehicles, less requests need to be subcontracted. When clustered requests are subcontracted, the requests are assigned to taxis separately. We assume the following conditions which have to be satisfied for requests to be clustered.

- The capacity of the vehicle is not allowed to be violated. Based on this condition, the number of requests that can be clustered depends on the number of customers, the type of the customer(s), and the type of the vehicle used.
- The time window specified by the customer(s), either for pickup or delivery, is not allowed to be violated.
- The excess ride time of each request (in the literature also known as the maximum ride time) does not exceed a certain bound. The excess ride time is defined as the difference between the actual ride time and the direct ride time of transporting the customer(s) directly from the origin to the destination.

A group of clustered requests is called a route.

3 Two-stage recourse model

In the DaPP problem, a decision is made based on early requests, and under uncertainty of late requests. Besides this uncertainty, clustering is an important aspect. To incorporate these important features of the problem in the model, we have built a non-standard two-stage recourse model. In the first stage, all early requests are considered, while the second-stage problem also considers the late requests. This is consistent with most two-stage recourse models used in stochastic programming. Our model is non-standard since each stage consists of two consecutive optimization problems. The first optimization problem is the clustering of requests into routes and can be viewed as preprocessing. The second optimization problem deals with the assignment of the obtained routes to vehicles (own vehicles and taxis). For more information about recourse models and stochastic programming, we refer the interested reader to Birge and Louveaux (1997), Kall and Mayer (2005), Ruszczynski and Shapiro (2003).

We now present a schematic overview of our two-stage recourse model. Subsequently, we introduce some notation and give a more detailed description.

Figure 1 contains a schematic overview of the model. The first stage models the day before the day of operation. In this stage, all early requests are clustered into routes and these routes are in turn assigned to vehicles. The requests in the routes that are assigned to taxis are handed over to subcontractors and disappear from the system. Hence, these routes are not considered anymore, whereas the routes assigned to own vehicles are reconsidered in the second stage which models the day of operation. In this second stage, the late requests are clustered into routes together with the routes assigned to own vehicles in the first stage. Finally, these routes are assigned to an own vehicle (car or van), or to a taxi.

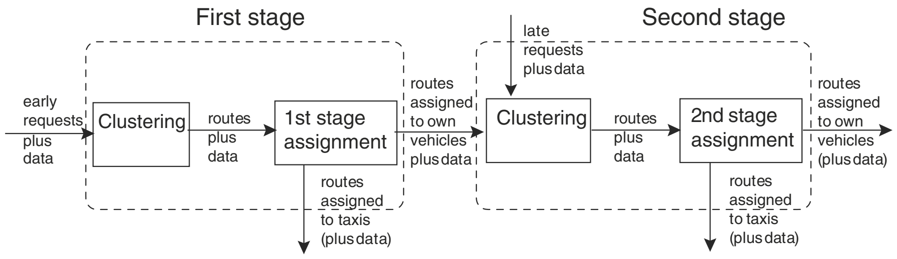

Fig. 1 Schematic overview of the two-stage recourse model

The following notation is used.

Sets and indices

I^k	routes of stage k, index i, $k = 1, 2$
I_v^k	van routes of stage k, $I_v^k \subset I^k$, $k = 1, 2$
J	own vehicles, index j, $J = J_v \cup J_c$, $J_v \cap J_c = \emptyset$
J_v	own vans
J_c	own cars
T	time periods, index t

Here, I^1 is the set of early requests clustered into routes, and I^2 is the set of early routes assigned to own vehicles in the first stage and the set of late requests clustered together into new routes.

Van routes are routes in which one or more wheelchair customers need to be transported. These routes need to be served with a van, or subcontracted.

Time is discretized in periods. All time periods have an equal length, which is measured in minutes.

Parameters

b_i	time period in which route i starts
e_i	time period in which route i ends, $e_i \geq b_i$
a_t^v	number of vans available during time period t
a_t^c	number of cars available during time period t
c_i^a	cost of assigning route i to a taxi
c_i^v	cost of serving route i with an own van
c_i^c	cost of serving route i with an own car
β	cost multiplier for short-term subcontracting, $\beta > 1$

Here, c_i^v and c_i^c depend on the distance of the route, and c_i^a consists of the costs of assigning the requests in the route separately to subcontractors. Hence, c_i^a depends on the number of requests in the route and on the distance of each request in the route. The cost multiplier β is used to model that subcontracting on the day of operation is more expensive than day-ahead.

First-stage decision variables (all binary)

x_i	1 if route i is assigned to a taxi, 0 if the route is assigned to an own vehicle
x_{ij}	1 if route i is assigned to own vehicle number j, 0 otherwise
x_{ijt}	1 if route i is assigned to own vehicle number j in time period t, 0 otherwise

Second-stage decision variables (all binary)

y_i	1 if route i is assigned to a taxi, 0 if the route is assigned to an own vehicle
y_{ij}	1 if route i is assigned to own vehicle number j, 0 otherwise
y_{ijt}	1 if route i is assigned to own vehicle number j in time period t,0 otherwise

Furthermore, we introduce the matrix ω of random variables, whose realizations become known only in the second stage. The matrix ω contains all information on the late requests. Each row of the matrix contains the data of precisely one such request. The number of rows, i.e., the number of late requests, is random too.

Description of the recourse model In the first stage, all early requests are clustered into routes. In Sect. 4.1, a description is given of the clustering heuristic which is applied to combine requests into routes.

For the assignment of routes to vehicles, we have developed a model which is related to the Multi-Resource Generalized Assignment Problem (MRGAP) (see e.g. Gavish and Pirkul 1991; Mazzola and Wilcox 2001). MRGAP is a generalized assignment problem in which agents may use different, limited resources to perform the jobs assigned to them. The resemblances between our model and MRGAP are the assignment of routes (jobs) to vehicles (agents), and the existence of different vehicle types (resources). The most important difference, which destroys the special structure of MRGAP, is the use of time periods in our model. Other differences are the cost structure and the resource taxi of which we assume an unlimited number is available.

The mathematical formulation of our assignment model is as follows.

$$\min \sum_{i \in I^1} c_i^a \cdot x_i + Q(x) \tag{1}$$

$$\begin{aligned} \text{s.t.} \sum_{i \in I^1, j \in J_v} x_{ijt} &\leq a_t^v, \quad t \in T \\ \sum_{i \in I^1, j \in J_c} x_{ijt} &\leq a_t^c, \quad t \in T \end{aligned} \tag{2}$$

$$x_i = 1 - \sum_{j \in J} x_{ij}, \quad i \in I^1 \tag{3}$$

$$x_{ijt} = \begin{cases} x_{ij} & t \in [b_i, e_i] \\ 0 & \text{otherwise} \end{cases}, \quad i \in I^1, \quad j \in J \tag{4}$$

$$x_{ij} = 0, \quad i \in I_v^1, \quad j \in J_c \tag{5}$$

$$\begin{aligned} & x_i \in \{0,1\}, \quad i \in I^1 \\ & x_{ij} \in \{0,1\}, \quad i \in I^1,\ j \in J \\ & x_{ijt} \in \{0,1\}, \quad i \in I^1,\ j \in J,\ t \in T \end{aligned} \tag{6}$$

The objective function (1) minimizes the costs of assigning the early routes to taxis plus $Q(x)$, which is the expected costs of the second-stage problem resulting from the first-stage (assignment) decision x. The constraints (2) ensure that in each time period no more routes are assigned to cars (vans) than available. The constraints (3) are a definition statement: each route that is not assigned to an own vehicle is assigned to a taxi. The constraints (4) ascertain that if a route is assigned to a certain vehicle, this vehicle will serve the entire route. The constraints (5) ensure that all van routes are assigned to a van or to a taxi. The last set of constraints (6) defines all variables as binary variables.

Let us take a closer look at $Q(x)$, the expected costs of the second-stage problem associated with the first-stage decision x. Although the vector x contains all first-stage decisions, only the routes which are assigned to own vehicles in the first stage are reconsidered in the second stage. The expectation is taken with respect to ω, the matrix of random variables with all information on the late requests. Suppose a particular realization of the late requests is given. Then, these requests are clustered, together with the routes assigned to own vehicles in the first stage, by applying the clustering heuristic described in Sect. 4.1. New routes are obtained from this clustering which can be viewed as preprocessing of the second-stage assignment. The assignment of these routes to vehicles is solved by the following model.

$$\min \sum_{i \in I^2} \beta \cdot c_i^a \cdot y_i + \sum_{i \in I^2, j \in J_v} c_i^v \cdot y_{ij} + \sum_{i \in I^2, j \in J_c} c_i^c \cdot y_{ij} \tag{1'}$$

$$\begin{aligned} \text{s.t.} \sum_{i \in I^2, j \in J_v} y_{ijt} \le a_t^v, \quad t \in T \\ \sum_{i \in I^2, j \in J_c} y_{ijt} \le a_t^c, \quad t \in T \end{aligned} \tag{2'}$$

$$y_i = 1 - \sum_{j \in J} y_{ij}, \quad i \in I^2 \tag{3'}$$

$$y_{ijt} = \begin{cases} y_{ij} & t \in [b_i, e_i] \\ 0 & \text{otherwise} \end{cases}, \quad i \in I^2, \quad j \in J \tag{4'}$$

$$y_{ij} = 0, \quad i \in I_v^2,\ j \in J_c \tag{5'}$$

$$\begin{aligned} &y_i \in \{0, 1\}, \quad i \in I^2 \\ &y_{ij} \in \{0, 1\}, \quad i \in I^2,\ j \in J \\ &y_{ijt} \in \{0, 1\}, \quad i \in I^2,\ j \in J,\ t \in T \end{aligned} \tag{6'}$$

The structure of this model is very similar to the structure of the model used in the first stage: only the objective function is different. In the first stage, the assignment of routes to taxis is permanent and the assignment of routes to own vehicles is preliminary. Consequently, the objective function consists only of the costs for subcontracting. In the second stage, all assignments are permanent. Hence, the objective function comprises the costs of all assignments. Furthermore, subcontracting on the day of operation is more expensive than subcontracting one day ahead.

We conclude this section with a mathematical formulation for $Q(x)$. First of all, we denote the optimal value of the model (1′)–(6′) by v. Amongst others, v depends on the data D of the routes that are under consideration. D can be represented as a matrix which results from applying the clustering heuristic to two sets of routes/requests. The first set is the set of routes that have been assigned to own vehicles in the first stage. This depends on the vector x. The second set consists of the realization of the late requests, which is the realization of the random variables of the matrix ω. To make the dependence of the matrix D on x and ω visible, we write $D(x, \omega)$. Then

$$Q(x) = E_\omega[v(D(x, \omega))].$$

4 Solution method

In the previous section, the two-stage recourse model for the DaPP problem has been explained in detail. The model consists of two stages, and each stage consists of two optimization problems: the clustering of requests into routes, and the assignment of the obtained routes to vehicles. In this section, we present the heuristics we have developed to solve each optimization problem: in Sect. 4.1 the clustering heuristic is presented and in Sect. 4.2 the assignment heuristic. Furthermore, in Sect. 4.3 we describe the genetic algorithm to solve the two-stage recourse model as a whole.

The reason that we have developed our own clustering heuristic instead of using one of the heuristics described in the literature (e.g. Borndörfer et al. 1997; Desrosiers et al. 1991; Ioachim et al. 1995) is that the CPU time of the heuristics is very important. To evaluate $Q(x)$ for a fixed first-stage decision x, both the clustering and assignment problem have to be solved many times. As a result, the CPU time to evaluate $Q(x)$ depends heavily on the CPU time of both heuristics. The clustering heuristic we have developed uses special characteristics of our problem and appears to be very fast.

The assignment heuristic which we have developed is partly based on heuristics in the paper of Dawande et al. (2000).

Since genetic algorithms have been applied successfully to problems related to our assignment problem, see Baker and Ayechew (2003), Chu and Beasley (1997), Chu and Beasley (1998), Feltl and Raidl (2004), we have decided to use a genetic algorithm to solve the recourse model.

4.1 Clustering heuristic

Our clustering heuristic is based on the data of individual requests. To allow application of the heuristic in the second stage, in which late requests are combined together with the routes assigned to own vehicles in the first stage, the routes therefore are reformulated as requests. This allows to use the same heuristic in both stages: in the first stage to combine early requests, and in the second stage to combine reformulated routes and late requests.

The purpose of the clustering heuristic is to cluster all requests in as few routes as possible, i.e., each route consists of at least one request, but preferably more. Our clustering heuristic is a so-called greedy heuristic, and hence the order in which the requests are considered is important. We choose to order the requests based on the direct ride time of the request. Requests with a long direct ride time have a good potential to be clustered with many other requests.

We start the clustering heuristic by splitting the group of requests into subgroups, such that either the pickup and/or the delivery location of all requests in a subgroup are the same. The first subgroups consist of requests which have as origin and/or destination a special location. Since these special locations are more likely to occur than other locations, we expect better combination possibilities. All requests that have no location in common will also be put in a subgroup.

In each subgroup, the requests are ordered in decreasing direct ride time; ties are broken using increasing start time of the pickup time window, in remaining cases the order is arbitrary.

For each subgroup, we take the first van request. Since the capacity of a van is greater than the capacity of a car, in potential more requests can be clustered with a van request. Now, the other requests are tried to be clustered with this van request, in the order specified by the list. For this, the clustering conditions have to be satisfied. As described in Sect. 2, the capacity of the vehicle is not allowed to be violated, the time window for pickup or delivery has to be satisfied, and the excess ride time does not exceed a certain bound. If all requests are checked for clustering with the van request, the clustered requests form a route, and the data of this new route is calculated. The data of a route consists of an indicator of the vehicle type (a van route or not), the three different cost parameters of serving the route with a car, van, and a taxi, respectively, and the time periods in which the route starts and ends. In the clustering heuristic, time is measured continuously, unlike the assignment heuristic in which time is discretized in periods. Hence, the begin and end time of the route are converted into a begin and end period, respectively. Finally, the requests in the route are deleted from the subgroup. This process is repeated until there are no more van requests in the subgroup.

Next, the remaining (non-van) requests in the subgroup are combined into routes, following the same steps as described above.

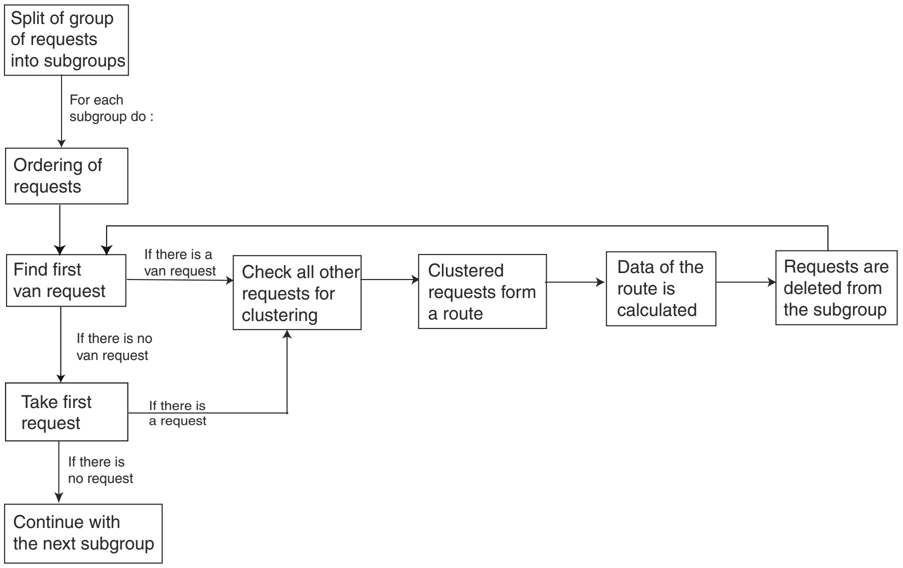
Split of group of requests into subgroups
For each subgroup do :
Ordering of requests
Find first van request
If there is a van request
Check all other requests for clustering
Clustered requests form a route
Data of the route is calculated
Requests are deleted from the subgroup
If there is no van request
Take first request
If there is a request
If there is no request
Continue with the next subgroup

Fig. 2 Flow chart of the clustering heuristic

This way, all subgroups are handled. In Fig. 2, a flow chart of the clustering heuristic can be found.

4.2 Assignment heuristic

The purpose of the assignment heuristic is to find a feasible assignment of all routes to vehicles (own vehicles and taxis) such that the costs are as low as possible. We propose a greedy heuristic which considers the routes one-by-one. The route under consideration is assigned to an own vehicle if it passes the feasibility test, and to a taxi if not. Here we use a relaxed feasibility test: for assigning a route to a car (van), the individual cars (vans) are not considered as is done in our model in Sect. 3 (by use of the variables x_{ij} and y_{ij}). Instead, for each time period it is checked whether the number of available cars (vans) is large enough to serve all routes already assigned plus the route under consideration. Our assignment neglects scheduling aspects but is suitable for the decision we want to make, i.e., which routes (requests) are best to assign to taxis.

Because our heuristic is greedy, the ordering of the routes is important. For the ordering of routes we use the difference in costs between subcontracting and using an own vehicle.

Let c_i^{diff} the difference in costs for route i and

$$c_i^o = \begin{cases} c_i^v & \text{if route } i \text{ is a van route} \\ c_i^c & \text{otherwise} \end{cases},$$

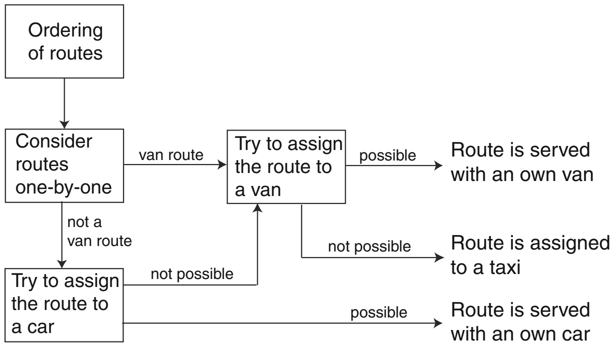
Ordering of routes
Consider routes one-by-one
van route
Try to assign the route to a van
possible
Route is served with an own van
not a van route
not possible
Route is assigned to a taxi
Try to assign the route to a car
not possible
possible
Route is served with an own car

Fig. 3 Flow chart of the assignment heuristic

```
Generation of initial population;
Evaluation of fitness of individuals in the population;
repeat:
    Selection of parents from the population;
    Reproduction of parents to create children;
    Evaluation of fitness of children;
    Replacement of (part of) the population by the children;
until   stopping criterion is satisfied.
```

Fig. 4 The main steps of a genetic algorithm

then $c_i^{\text{diff}} = c_i^a - c_i^o$. Here, the symbols c_i^v, c_i^c, and c_i^a are defined in Sect. 3. The higher the cost difference the better it is to serve the route with an own vehicle.

The assignment heuristic starts by sorting all routes in decreasing order of cost difference; ties are broken using increasing order of time period in which the routes start, in remaining cases the order is arbitrary.

Working down the list of routes, they are assigned to vehicles by the following rules:

1. Van routes are assigned to vans if possible, otherwise they are assigned to taxis.
2. Non-van routes are assigned to cars if possible. If not, they are assigned to vans or—if that fails too—to taxis.

Figure 3 contains a flow chart of the assignment heuristic.

4.3 Genetic algorithm

Genetic Algorithms (GAs) were developed by Holland (1975). Although nowadays their principles are well known, we have summarized the main steps of a GA in Fig. 4. For more details and terminology used we refer the interested reader to e.g. Goldberg (1989) and Reeves (1993).

A GA starts with the generation of an initial population of individuals. Each individual is represented by a string of characters (or bits) representing a possible solution to the given problem. The fitness of an individual is determined by use of a given objective function. In each iteration, individuals reproduce (or mate) in a crossover procedure to create children (or offspring). Selection ensures that mostly highly fit individuals are chosen to mate. After the crossover procedure, mutation is often applied to the offspring by changing some characters in the string. The offspring can either replace the whole population, or only the less fit individuals. This selection-reproduction-evaluation-replacement cycle ends when the stopping criterion is satisfied.

When developing a GA, many choices have to be made regarding for example the selection procedure, crossover procedure, and stopping criterion. However, a suitable representation needs to be chosen first. This is the way individuals in the population are represented. We have chosen to use a standard 0-1 binary representation: for each route a binary variable indicates whether the route is assigned to an own vehicle (zero) or to a subcontractor (one). Thus, a member of the population is an n-bit string with n the number of routes in the first stage. Consequently, there are 2^n possible distinct members, or solutions. Not all solutions are feasible. A solution is said to be infeasible if in any time period more own vehicles of a particular type are needed than there are available. Infeasible solutions are discarded and will never enter the population.

The fitness value of an individual is determined by the fitness function. We want to minimize the fitness value of the individuals, i.e., we want to obtain the least fit individual. Since this is counterintuitive, we have decided to use the term cost function instead of fitness function and the term estimated costs instead of fitness value. The cost function is equal to the objective function of the (first-stage) assignment model. Let x be the binary vector as defined before, then the cost function is

$$\widehat{f}(x) = \sum_{i=1}^{n} c_i^a \cdot x_i + \widehat{Q}(x).$$

To determine $\widehat{Q}(x)$ a sample of the random variables of the matrix ω is drawn. Let s be the sample size, and ω_j the realization of the late requests of sample j, then

$$\widehat{Q}(x) = \frac{1}{s} \sum_{j=1}^{s} v(D(x, \omega_j)).$$

To solve the second-stage problem for a particular realization and given first-stage solution the clustering and assignment heuristic have to be applied. For a large sample size, i.e., many realizations, calculating the estimated costs of an individual will take much time. Using only a small number of realizations might give a bad approximation of $Q(x)$ and thus of $f(x)$. In Sect. 5.2 we will decide about the number of realizations to use and report on the error encountered.

Parents are selected by the binary tournament selection method. In this method, two individuals of the population are chosen randomly. The cheapest individual of the two is chosen as the first parent. The second parent is obtained in the same manner.

We use the one-point crossover in which a crossover point is selected randomly. The first child is created by taking the bits to the left of this point from the first parent, and by taking the other bits from the second parent. Swapping parents gives the second child, i.e., the second child obtains the complementary bits from both parents.

After the crossover procedure two bits in each child are randomly mutated. Then, each child is checked for feasibility. Even with the relaxed feasibility introduced in Sect. 4.2, checking for feasibility is not easy because the non-van routes can be assigned to a car or to a van. We have decided to perform a simple check for relaxed feasibility which never accepts an infeasible solution, but might reject a (just) feasible solution.

Replacement occurs according to the steady-state replacement method. In this method, children replace the most expensive population member, if the estimated costs of the child are lower. Duplicate solutions are not allowed to enter the population.

5 Numerical experiments

In this section, we describe our numerical experiments and the results: in Sect. 5.1 we describe how the data instances are generated and in Sect. 5.2 the parameters of the GA are discussed. Furthermore, in Sect. 5.3 the effect of taking into account the late requests can be found, and in Sect. 5.4 the effect of clustering the requests into routes.

5.1 Data

All data instances used for the experiments are randomly generated and contain 50 early requests. The number of late requests varies between 5 and 25. The pickup time is between 7 and 10 am, and the delivery time is until noon. All requests consist of one customer. Each request has a chance of 0.125 of being a van request which needs to be served with a van because of a wheelchair customer. The capacity of a car (van) is equal to 3 (6). Until 10 am 10 cars are available, thereafter 4, and all day 3 vans are available. All locations lie on a 7×7-grid and have an equal chance of being chosen, except for two special locations, e.g., hospitals, which have a higher chance. This chance varies between 0.05 and 0.3. The time periods have a length of 15 min. The costs of serving a request/route for one grid unit with a car, van, and taxi are, respectively, 0.8, 1, and 1. Furthermore, for each subcontracted request a fixed fee of 3 needs to be paid, and $\beta = 1.2$. All experiments are performed on a 2.4 Ghz Intel Pentium 4. With the parameter setting of Sect. 5.2, the running time per instance varies between 60 and 120 min of CPU time.

5.2 Parameters of the GA

The size of the population is set equal to 30 members. We have experimented with both random as well as structured initial population members (see below), and observed that the latter leads to faster convergence of the GA to a good solution. Therefore, we only use structured initial population members which are obtained by diminishing the number of available own vehicles for certain time periods, or by setting a maximum on the number of routes that are allowed to be served with an own vehicle, or by doing both. That is, a structured initial solution reserves capacity today to be able to

Table 1 The estimated costs of the GA solution and the 95% confidence interval, for four different instances

Estimated costs	Lower bound	Upper bound	Length (in percentage)
250.48	248.42	252.53	1.65
341.62	338.52	344.72	1.83
284.22	281.71	286.72	1.78
212.26	210.21	214.31	1.95

serve requests that become known tomorrow. For each population member, a different setting to reserve capacity is used, e.g., for the first population member at most 90% of the routes are allowed to be served with own vehicles and for the second population member this is at most 50%. If two structured initial solutions are the same, a randomly generated initial solution is used instead of the duplicate one. Such solutions are constructed by randomly choosing routes which are assigned to own vehicles up to the available number for each vehicle type.

As stopping criterion we let the GA run for 500 generated non-duplicate feasible children. Based on tests with various instances, it appears that after 500 non-duplicate feasible children have been generated the solution improves only marginally.

Now, only the sample size has to be determined. In Sect. 4.3 it has already been mentioned that the number of realizations influences the quality of the approximation of the cost function $f(x)$. To measure the error encountered by using a sample for the late requests, a confidence interval for the optimal value of the cost function $f(x)$ can be determined. In Table 1, we have denoted for four different instances the estimated costs of the solution and the 95% confidence interval when 200 realizations are used. The length of the confidence interval is less than 2% for all four instances. Since we think that this is reasonable, we set the sample size equal to 200 realizations.

A second factor influencing the quality of the results is the randomness of the algorithm itself: possibly good solutions may not always be found. To test this effect, we ran the algorithm ten times on the same instance, and did this for various instances. As we found that the estimated costs vary by at most 1.03%, we conclude that the algorithm consistently finds similar solutions.

5.3 Late requests

To determine the effect of taking into account the late requests (by using a two-stage recourse model), we introduce the myopic solution in which the late requests are ignored. That is, the myopic solution is obtained by clustering the early requests and assigning the obtained routes to vehicles. The estimated costs of the myopic solution are calculated by the cost function $\widehat{f}(x)$ defined before.

For a good comparison of both approaches, we generated 12 instances which are solved twice: once with few (5–10) late requests, and once with many (20–25) late requests. Since the early requests are the same for each variant, by definition the myopic solution is the same for few and many late requests, only the estimated costs of the myopic solution are different. For each instance the estimated costs of the GA

Table 2 The effect of late requests on the estimated costs of the GA solution and the myopic solution

Inst.	Few late requests			Many late requests		
	GA solution	Myopic solution	Difference in percentage	GA solution	Myopic solution	Difference in percentage
1	222.04	229.00	3.14	312.91	322.80	3.16
2	263.22	277.00	5.24	358.82	378.65	5.53
3	261.76	267.07	2.03	360.14	370.62	2.91
4	250.48	256.57	2.43	345.20	362.59	5.04
5	252.56	264.06	4.55	352.58	370.04	4.95
6	257.98	263.81	2.26	348.03	364.73	4.80
7	236.99	259.92	9.67	341.62	366.03	7.15
8	224.74	229.71	2.21	319.09	328.92	3.08
9	232.23	241.53	4.00	329.83	344.33	4.39
10	265.85	271.55	2.14	363.18	373.12	2.74
11	274.42	285.18	3.92	366.61	379.37	3.48
12	246.23	258.04	4.79	340.05	356.00	4.69

solution and the myopic solution are calculated, both with few and many late requests. Furthermore, the relative difference between the estimated costs of the GA solution and myopic solution is determined.

As can be seen in Table 2, the estimated costs of the myopic solutions are between 2 and 10% higher than those of the GA solutions. This is as we expected and can be explained as follows. The myopic solution simply tries to assign as few early requests as possible to subcontractors. Hence, on the day of operation there is hardly any capacity of own vehicles left and many requests have to be assigned to subcontractors, which is 20% more expensive than the day before. The GA solution does take into account the late requests. As a result, more early requests will be assigned to subcontractors in the first stage, but on the day of operation itself less requests need to be subcontracted. When many late requests arise, this effect should be and is almost always stronger, as can be seen in Table 2: the difference (in percentage) between the estimated costs of the GA solution and myopic solution is larger than when few late requests arise, except for instances 7, 11, and 12.

Table 3 clearly demonstrates the effect explained above. The myopic solution assigns as many early requests as possible to own vehicles while the GA solution reserves capacity for the late requests. The columns costs show the estimated costs of the GA solution and the myopic solution when few and many late requests are used, and can also be found in Table 2. The remaining columns show the costs of the first stage, i.e., the costs of subcontracting the early requests, and the number of early requests assigned to subcontractors. Only for instance 12 with few late requests, the GA solution assigns less early requests to subcontractors than the myopic solution. This is caused by an unfortunate outcome of the assignment heuristic.

As can be seen in Table 3, when many late requests arise the effects are the same as when few late requests arise, but only larger. Considerably more early requests are

Table 3 The effect of late requests on the GA solution and the myopic solution

Inst.	Few late requests						Many late requests					
	GA solution			Myopic solution			GA solution			Myopic solution		
		Subcontracted			Subcontracted			Subcontracted			Subcontracted	
	Costs	Costs	Number	Costs	Costs	Number	Costs	Costs	Number	Costs	Costs	Number
1	222.04	67	9	229.00	24	3	312.91	67	9	322.80	24	3
2	263.22	93	11	277.00	65	10	358.82	160	19	378.65	65	10
3	261.76	90	11	267.07	65	9	360.14	153	18	370.62	65	9
4	250.48	74	10	256.57	46	7	345.20	123	15	362.59	46	7
5	252.56	94	11	264.06	56	8	352.58	141	17	370.04	56	8
6	257.98	83	11	263.81	64	10	348.03	133	15	364.73	64	10
7	236.99	66	7	259.92	46	7	341.62	96	12	366.03	46	7
8	224.74	67	8	229.71	49	7	319.09	78	9	328.92	49	7
9	232.23	77	9	241.53	62	9	329.83	118	15	344.33	62	9
10	265.85	99	12	271.55	78	12	363.18	136	17	373.12	78	12
11	274.42	102	11	285.18	73	10	366.61	123	15	379.37	73	10
12	246.23	77	9	258.04	67	10	340.05	120	15	356.00	67	10

assigned to subcontractors in the GA solution, and hence the first-stage costs of the GA solution are also larger.

5.4 Clustering

The second important aspect of the DaPP problem, besides the uncertainty of the late requests, is the clustering of requests into routes. To determine the effect of clustering, we have developed the NoClust solution for comparison. This solution is obtained by applying the GA to individual requests—as opposed to clustered routes—in both stages. The estimated costs of the NoClust solution are obtained by the costs of the early requests that are assigned to subcontractors in the first stage and the estimated costs of the assignment of the remaining early requests and all late requests to vehicles (both own vehicles and subcontractors) in the second stage.

Again, we generated 12 instances each with two variants: few and many clustering possibilities, implemented by specifying the probability of a special location as 0.05 and 0.3, respectively. These probabilities hold for both the early and the late requests. For each instance, the estimated costs of the GA solution and NoClust solution are calculated, as well as their relative difference.

As can been seen in Table 4, the estimated costs of the NoClust solution are between 10 and 81% higher than the estimated costs of the GA solution. As expected, the difference is largest when there are many clustering possibilities. But even when there are only few clustering possibilities the difference in estimated costs is considerable.

Table 4 The effect of clustering on the estimated costs of the GA solution and NoClust solution

Instance	Limited clustering			Much clustering		
	GA solution	NoClust solution	Difference (%)	GA solution	NoClust solution	Difference (%)
1	296.45	335.61	13.21	173.37	287.09	65.60
2	281.69	330.50	17.33	174.37	315.85	81.14
3	284.22	321.59	13.15	232.69	335.34	44.12
4	299.42	339.04	13.23	195.73	333.61	70.45
5	289.31	337.32	16.60	219.06	353.73	61.48
6	258.98	323.04	24.73	184.64	301.37	63.22
7	266.93	315.01	18.01	217.78	348.37	59.97
8	283.76	310.74	9.51	188.81	310.60	64.50
9	249.37	302.58	21.34	215.53	335.98	55.88
10	263.13	319.61	21.46	222.31	326.48	46.85
11	288.77	334.67	15.89	212.26	348.04	63.96
12	240.56	291.94	21.36	221.32	315.99	42.78

6 Conclusion

We have formulated the day-ahead paratransit planning problem. In this problem, in which we need to decide one day before the day of operation about the routes to assign to vehicles, the focus is on clustering requests into routes and and on coping with uncertainty due to the late requests that only become known just before the day of operation. In the non-standard two-stage recourse model we have developed, taxis allow for flexibility which is necessary to deal with the uncertainty. Our model and solution method are flexible and can easily be adjusted, e.g., to incorporate different numbers of customers, to have different distributions for the characteristics of the early and late requests, or to include other unforeseen events like canceled requests and vehicle breakdowns.

The results are promising. Clustering of requests into routes is very profitable: the estimated costs are much lower compared to the same instances when the individual requests are considered. Taking into account the late requests decreases the estimated costs with on average 4%. CPU times are, however, rather large. Further research will be necessary to decrease CPU times.

Another point for future research is changing the assumption that all late requests become known at the beginning of the day of operation. It is more realistic that they become known one-by-one during the day, i.e., that they are revealed in an online manner. Indeed, many studies (e.g. Attanasio et al. 2004; Cordeau and Laporte 2003a) point out the importance of incorporating look-ahead features. In our model, such a feature has already been included by considering the late requests. However, assuming that the late requests are revealed online will make the look-ahead feature more realistic. In a subsequent paper, this approach will be analyzed.

References

Attanasio A, Cordeau JF, Ghiani G, Laporte G (2004) Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Comput 30:377–387

Baker B, Ayechew M (2003) A genetic algorithm for the vehicle routing problem. Comput Oper Res 30(5):787–800

Birge J, Louveaux F (1997) Introduction to stochastic programming. Springer, New York

Borndörfer R, Grötschel M, Klostermeier F, Küttner C (1997) Telebus Berlin: vehicle scheduling in a dial-a-ride system. ZIB Report 97-23

Chu P, Beasley J (1997) A genetic algorithm for the generalized assignment problem. Comput Oper Res 24(1):17–23

Chu P, Beasley J (1998) A genetic algorithm for the multidimensional knapsack problem. J Heurist 4(1): 63–86

Cordeau JF, Laporte G (2003) The dial-a-ride problem (DARP): variants, modeling issues and algorithms. Q J Belgian French Ital Oper Res Soc 1:89–101

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

Dawande M, Kalagnanam J, Keskinocak P, Ravi P, Salman F (2000) Approximation algorithnms for the multiple knapsack problem with assignment restrictions. J Combin Optim 4:171–186

Desrosiers J, Dumas Y, Soumis F, Taillefer S, Villeneuve D (1991) An algorithm for mini-clustering in handicapped transport. Les Cahiers du GERAD, G-91-02, École des Hautes Études Commerciales, Montréal

Feltl H, Raidl G (2004) An improved hybrid genetic algorithm for the generalized assignment problem. In: Proceedings of the 2004 ACM symposium on applied computing, pp 990–995

Gavish B, Pirkul H (1991) Algorithms for the multi-resource generalized assignment problem. Manage Sci 37(6):695–713

Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading

Holland J (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press

Ioachim I, Desrosiers J, Dumas Y, Solomon M, Villeneuve D (1995) A request clustering algorithm for door-to-door handicapped transportation. Transport Sci 29:63–78

Jaw JJ, Odoni A, Psaraftis H, Wilson N (1986) A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transport Res Part B 20(3):243–257

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

Kall P, Mayer J (2005) Stochastic linear programming: models, theory, and computation. Kluwer, New York

Madsen O, Ravn H, Rygaard J (1995) A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Ann Oper Res 60:193–208

Mazzola J, Wilcox S (2001) Heuristics for the multi-resource generalized assignment problem. Naval Res Logist 48:468–483

Reeves C (1993) Modern heuristic techniques for combinatorial problems. Blackwell, New York

Ruszczynski A, Shapiro A (eds) (2003) Stochastic programming, handbooks in operations research and management science, vol 10. North-Holland, Amsterdam

Toth P, Vigo D (1997) Heuristic algorithms for the handicapped persons transportation problem. Transport Sci 31(1):60–71