Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://www.tandfonline.com/loi/tjor20

Solving the Dial-a-Ride problem using genetic algorithms

R M Jorgensen, J Larsen & K B Bergvinsdottir

To cite this article: R M Jorgensen, J Larsen & K B Bergvinsdottir (2007) Solving the Dial-a-Ride problem using genetic algorithms, Journal of the Operational Research Society, 58:10, 1321-1331, DOI: 10.1057/palgrave.jors.2602287

To link to this article: https://doi.org/10.1057/palgrave.jors.2602287

Published online: 21 Dec 2017.

Journal of the Operational Research Society (2007) 58, 1321–1331 © 2007 Operational Research Society Ltd. All rights reserved. 0160-5682/07 $30.00

www.palgrave-journals.com/jors

Solving the Dial-a-Ride problem using genetic algorithms

RM Jorgensen, J Larsen* and KB Bergvinsdottir

Technical University of Denmark, Lyngby, Denmark

In the Dial-a-Ride problem (DARP), customers request transportation from an operator. A request consists of a specified pickup location and destination location along with a desired departure or arrival time and capacity demand. The aim of DARP is to minimize transportation cost while satisfying customer service level constraints (Quality of Service). In this paper, we present a genetic algorithm (GA) for solving the DARP. The algorithm is based on the classical cluster-first, route-second approach, where it alternates between assigning customers to vehicles using a GA and solving independent routing problems for the vehicles using a routing heuristic. The algorithm is implemented in Java and tested on publicly available data sets. The new solution method has achieved solutions comparable with the current state-of-the-art methods.

Journal of the Operational Research Society (2007) **58,** 1321–1331. doi:10.1057/palgrave.jors.2602287
Published online 13 September 2006

Keywords: Dial-a-Ride; heuristics; specialized transportation

1. Introduction

In the Dial-a-Ride problem (DARP), customers request transportation from a transportation operator. A request consists of a specified pickup (origin) location and drop-off (destination) location along with a desired departure or arrival time and the number of passengers to be transported.

The problem consists of determining the best routing schedule for the vehicles, which minimizes overall transportation costs while maintaining a prescribed level of customer service. The service level estimation can be based on the ride times of the customers, deviations from desired departure or arrival times, etc. The challenge is to combine the conflicting factors: low cost of transportation *versus* a high level of service.

An example of a DARP transportation system is specialized transportation, that is, the transportation of children, disabled, elderly people, etc. These specialized transportations are usually provided by local government (see eg Borndorfer *et al*, 1997).

The main contribution of this paper is the demonstration that genetic algorithms (GAs) can be effectively implemented in a cluster-first, route-second approach to generate heuristic solutions to the DARP. The clustering is solved using the GA and the routing will be determined by a modified space–time nearest-neighbour heuristic presented by Baugh Jr *et al* (1998).

The solution method will be implemented in Java and tested using data sets described by Cordeau and Laporte (2003).

In the next section, the DARP will be described. A mathematical model will also be presented. Section 3 reviews related work, while our approach is presented in Section 4. Experimental results and the conclusions are presented in Section 5 and the overall conclusion is given in Section 6.

2. The DARP

The DARP is in the literature formulated in a number of different ways usually depending on the underlying real-life problem. The problem formulation here is focused on practical considerations present in the Danish transportation sector (see Jorgensen, 2002). The formulation is therefore based on the mathematical model of Jorgensen (2002).

In the Dial-a-Ride transportation system, we model customers that have to be transported from door to door, but not necessarily directly, that is, customers are allowed to share a ride but there are no fixed routes. This is, for example, the case in the transportation of elderly and disabled people in Denmark. All vehicles start and end their routes at a depot, but not necessarily the same depot.

We will consider the static case where all transportation requests (pickup and delivery) are known in advance. We define an upper limit on the number of vehicles available and assume that customers cannot be rejected. All vehicles have identical capacity C.

A time window is defined for all stops. These time windows are specified either by the customer or the transportation operator. The time windows are considered to be soft

Correspondence: J Larsen, Informatics and Mathematical Modelling, Richard Petersens Square 1, Technical University of Denmark, 2800 Lyngby, Denmark.
E-mail: jla@imm.dtu.dk

time windows. Soft time windows are useful when evaluating the tradeoffs between service requirements and cost requirements. Solutions with soft time windows indicate the degree of violation, thus allowing penalty methods to distinguish between a given pair of infeasible solutions in attempting to find a feasible region. The time windows are constructed based on the desired pickup or drop-off time given by the customer.

An upper bound on the length of the route duration, that is, the time it takes the vehicle to leave the depot, service all the customers on its route and return to the depot is set. If the maximum route duration is exceeded, it results in overtime pay to the driver or compensation by days off. Therefore, a violation is penalized in the objective function.

The cost in the DARP is calculated by a multi-objective function. The multi-objective function will be handled by combining the multiple objectives into one scalar objective minimizing the positively weighted sum of the objectives. The cost of transportation of the customers is estimated in this project to be the actual transportation cost and a 'cost of inadequate service'.

Transportation cost consists of transportation time which is defined as the total routing time of all the vehicles used in the transportation. The cost of inadequate service is defined by the excess ride time of customers and waiting time in the bus. Excess ride time is the extra time a customer is in the vehicle compared to a direct transportation from pickup to drop-off locations. The excess ride time gives a better estimate of the customer inconvenience than the total transportation time.

Constraints not included in the model are, for example, constraints concerning union rules and even distribution of customers on the routes. Distributing the customers evenly is desirable since it levels out the workload of the drivers. Costs not included are fixed costs such as capital cost, fixed costs for vehicles and depots, salary costs (assuming a fixed number of staff), etc.

Assume that we have a set of n customer requests. Each request specifies a pickup location, i, and delivery location, $n+i$. The customers also specify a demand, Δ_i, which is the number of seats required for the passengers that are to be transported from location i to $n+i$ at the same time, and either a preferred pickup time, a_i, or drop-off time, b_{n+i}. Each vehicle, k, starts at an origin depot $o(k)$ and ends at a destination depot $d(k)$ and each vehicle has a constant capacity C. Now we can define the following sets:

$P=\{1,\ldots,n\}$	set of pickup locations
$D=\{n+1,\ldots,2n\}$	set of delivery locations
$N=P\cup D$	set of pickup and delivery locations
K	set of vehicles
$V\subset K$	set of vehicles used in solution
$A=N\cup\{o(k),d(k)\}$	set of all possible stopping locations for all vehicles $k\in K$

We also define the following parameters:

m	number of vehicles used in the solution, that is, $\|V\|=m$
a_i	earliest time that service is allowed to start at in location i
b_i	latest time that service is allowed to start at in location i
s_i	service time needed at location i
$t_{i,j}$	travelling time or distance from location i to j
l_i	change in load at location i
r^k	maximum route duration for vehicle k
u_i	maximum ride time for a customer with pickup at location i

The following decision variables will be used in the model:

$$x_{i,j}^k=\begin{cases}1 & \text{if vehicle } k \text{ services a customer at location } i \\ & \text{and the next customer at location } j\\ 0 & \text{otherwise}\end{cases}$$

T_i^k	time at which vehicle k starts its service at location i
L_i^k	load of vehicle k after servicing location i
W_i^k	waiting time of vehicle k before servicing location i.

In the model, the weights in the objective function will be the following:

w_1	weight on customers transportation time
w_2	weight on excess ride time
w_3	weight on waiting time for customers
w_4	weight on work time
w_5	weight on time window violation
w_6	weight on excess of maximum ride time
w_7	weight on excess work time

The resulting mathematical model then becomes:

$$\begin{aligned}\min\quad & w_1\sum_{k\in V}\sum_{i,j\in A} t_{i,j}x_{i,j}^k + w_2\sum_{k\in V}\sum_{i\in P}(T_{n+i}^k - s_i - T_i^k - t_{i,n+i})\\ & + w_3\sum_{k\in V}\sum_{i\in N} W_i^k(L_i^k - l_i) + w_4\sum_{k\in V}(T_{d(k)}^k - T_{o(k)}^k)\\ & + w_5\sum_{k\in V}\sum_{i\in A}\max(0, a_i - T_i^k, T_i^k - b_i)\\ & + w_6\sum_{k\in V}\sum_{i\in P}\max(0, T_{n+i}^k - T_i^k) - u_i)\\ & + w_7\sum_{k\in V}\max(0, (T_{d(k)}^k - T_{o(k)}^k) - r^k)\end{aligned} \tag{1}$$

subject to

$$\sum_{k\in V}\sum_{j\in P\cup d(k)} x^k_{o(k),j} = m \qquad (2)$$

$$\sum_{k\in V}\sum_{i\in D\cup o(k)} x^k_{i,d(k)} = m \qquad (3)$$

$$\sum_{j\in A} x^k_{i,j} - \sum_{j\in A} x^k_{j,i} = 0 \qquad \forall k \in V,\ i \in N \qquad (4)$$

$$\sum_{k\in V}\sum_{j\in N} x^k_{i,j} = 1 \qquad \forall i \in P \qquad (5)$$

$$\sum_{j\in N} x^k_{i,j} - \sum_{j\in N} x^k_{j,n+i} = 0 \qquad \forall k \in V,\ i \in P \qquad (6)$$

$$x^k_{i,j}(T^k_i + s_i + t_{i,j} + W^k_j - T^k_j) \leqslant 0 \quad \forall k\in V,\ i, j\in A \quad (7)$$

$$T^k_i + s_i + t_{i,n+i} + W^k_j - T^k_{i+n} \leqslant 0 \quad \forall k\in V,\ i\in P \qquad (8)$$

$$x^k_{i,j}(L^k_i + l_j - L^k_j) = 0 \qquad \forall k \in V,\ i, j \in A \qquad (9)$$

$$l_i \leqslant L^k_i \leqslant C \qquad \forall k \in V,\ i \in P \qquad (10)$$

$$L^k_{o(k)} = L^k_{d(k)} = 0 \qquad \forall k \in V \qquad (11)$$

$$x^k_{i,j} \in \{0, 1\} \qquad \forall k \in K,\ i, j \in A \qquad (12)$$

$$T^k_i \geqslant 0 \qquad \forall k \in K,\ i \in V \qquad (13)$$

$$L^k_i \geqslant 0 \qquad \forall k \in K,\ i \in V \qquad (14)$$

$$W^k_i \geqslant 0 \qquad \forall k \in K,\ i \in V \qquad (15)$$

The objective function (1) of the DARP is a multi-criteria objective function. The objective function consists of the competing objectives of minimizing the total transportation cost, and the inconvenience to the customers. The total transportation cost is estimated to be proportional to the total time used when transporting the customers by all the vehicles, the total number of vehicles used in the solution, and the total route time of all vehicles used. Customer inconvenience is estimated to be proportional to the total excess ride time for the customers and the total waiting time for the customers in the vehicles.

In order to handle this multi-criteria objective function, each part of the objective function is multiplied by a weight. These weights are denoted $w_1, w_2, \ldots, w_7$. The values of the weights are then used to decide the relative weight of each criteria in the overall problem.

The depot constraints (2) and (3) describe the requirement that each vehicle starts and ends in a depot. The constraints allow a vehicle to leave an origin depot and drive straight to a destination depot without servicing any customers. The routing constraints (4) simply state that all locations must be visited.

The precedence constraints (5) and (6) represent the requirement that each customer must first be picked up at his pickup location and then dropped off at his delivery location by the *same* vehicle. The first of these constraints ensures that there is exactly one vehicle that leaves every pickup location, that is, every request is met. The second set of constraints states that the pickup and destination locations of a customer are serviced during the same trip.

Constraint (7) ensures that the arrival time at location $j(T^k_j - W^k_j)$ must be later than the sum of departure time from location i and the travelling time between the locations if that leg is to be part of the route.

To obtain a feasible solution, it is furthermore necessary to visit first the pickup point of a customer and then the delivery point. This is ensured by constraint (8).

Customers are divided into two groups: inbound and outbound. For inbound customers, the important issue is to arrive at the destination within some specified time window, and the time window at the pickup location is adjusted accordingly, whereas the time window for the pickup is given for the outbound customers and the time window at the destination is calculated accordingly.

The time windows for inbound customers are set according to the request of the customer regarding earliest pickup time. These desired times then set the lower limit for the pickup time window, that is, equal to a_i. Usually, the transportation operator or the authority specifies a time limit on the maximum deviation from these desired times, *dev*, usually 10–30 min. The upper limit on the pickup time window is set as: $b_i = a_i + dev$.

The lower limit on the delivery time window is the earliest possible arrival time, that is, the time at which the customer would arrive at the destination if picked up at the earliest pickup time, serviced, and transported directly from the pickup to destination, that is: $a_{n+i} = a_i + s_i + t_{i,n+i}$.

The upper limit is set as the latest feasible arrival time for the customer, for which the limits on ride time and pickup time are observed. The ride time constraint can be formulated using the concept of maximum excess ride time, which is the difference between actual ride time and direct transportation time. Usually, an upper limit on the excess ride time, E, is specified for the customers. Excess ride time is the extra time the customer has to spend in the vehicle compared to being transported directly from pickup location to drop-off location. This excess time is often specified as a linear equation, for example, 5 min $+\frac{1}{2}$ (direct transport time). In this model, there is a limit on the total time each customer must spend in the vehicle. This allows setting different criteria for different customers. Then, the upper time limit for the drop off location becomes: $b_{n+i} = b_i + s_i + t_{i,n+i} + E$. These time window calculations are shown in Figure 1.

In the case of an outbound customer, the customer specifies the latest drop-off time, which is set as the upper limit on the drop-off time window, equal to b_{n+i}. The other time window

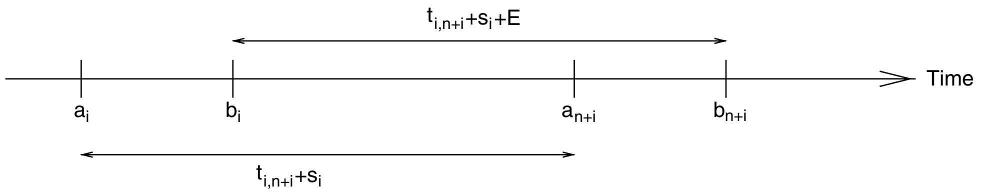

Figure 1 Setting the time windows for the drop-off location $[a_{n+i}, b_{n+i}]$ given the pickup locations time window $[a_i, b_i]$, the direct transportation time $t_{i,n+i}$ from i to $n+i$, service time s_i at i, and the upper limit on excess ride time E (figure from Cordeau and Laporte (2003)).

limits are then found using the same method backwards in time.

Constraints (10) ensure that the vehicle capacity is not exceeded.

The load of a vehicle at a point in time is the number of seats needed for the customers in the vehicle at that point in time. When a vehicle has serviced a pickup location i, the change in load is represented by $l_i = \Delta_i$ and the change in load after servicing a drop-off location $n+i$ is $l_{n+i} = -\Delta_i$. The actual load of vehicle k after servicing location i is L_i^k. Constraints (9) and (10) ensure that vehicle capacity is not exceeded and constraints (11) ensure that the actual loads of the vehicles are set to zero at the depots.

The actual arrival time of a customer at the destination location depends on different factors in the objective function, such as the time window violation, excess ride time, ride time violation, route duration, route duration violation, and waiting time.

If the customer arrives on time at his destination within the time windows, the cost contributed by the arrival of that customer is, if any, the excess ride time and the waiting time in an idle vehicle. On the other hand, if the customer is late, the time that passes from the upper bound of the time window to actual arrival is both the excess ride time and penalty for every minute the arrival time exceeds the upper time limit. If in addition the ride time of the customer exceeds the maximum ride time, a penalty is added to both the cost of excess ride time and time window violation. In this case, we assume that the maximum ride time for the customer plus the pickup time is higher than the upper bound on the drop-off time window. This case is presented in Figure 2. Thus, the cost of delivering a customer late increases in steps depending on the three constraints and of course on how late the customer is delivered.

If a vehicle arrives too early at a location, it has to wait. Two cases may arise: (1) no customers are present in the vehicle; or (2) one or more customers are waiting in the vehicle. In the first case, the minutes the vehicle has to wait add a penalty to the cost for violating the time windows, the route duration increases, and a route duration violation may occur. The reason for penalizing a time window violation even though the vehicle is empty is to have the same implementation of the time window violation. If there are customers present in

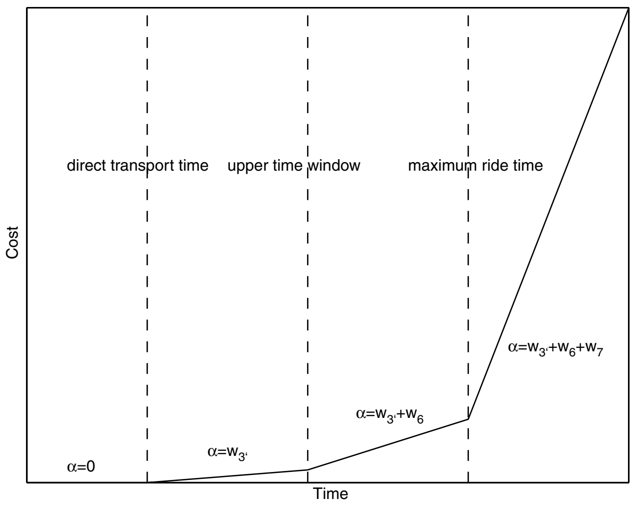

Figure 2 The influence of the arrival time on cost. The slope (α) of the cost increases in steps depending on the arrival time.

the vehicle, the waiting minutes will add to the cost in several ways: penalty for time window violation, the waiting time multiplied by the number of customers present in the idling vehicle, excess ride time for the customers, route duration increases, and possibly ride time or route duration violations as well.

The DARP can be proven to be $\mathcal{NP}$-hard (see eg Baugh Jr *et al*, 1998). The proof is based on the related $\mathcal{NP}$-hard traveling salesman problem with time windows, into which the DARP can be transformed.

3. Related work

This section contains a short description of previous closely related work within the field of the DARP.

The work by Jaw *et al* (1986) is generally considered pioneer research within DARP, and describes a sequential insertion heuristic algorithm. The objective function combines the minimization of operator costs and the minimization of customer inconvenience with respect to both customer ride time and deviation from desired pickup or drop-off times specified by each customer. The different parts of the objective function are balanced by multiplying with user-specified constants. In

Table 1 Overview of previous related work for the DAR problem

			Test results		
Paper	*Algorithm*	*Objectives*	*Size*	*Type*	*Time*
Jaw *et al* (1986)	Sequential insertion heuristic	Min operator costs, min customer ride times, min deviation from time windows	250	Random	20 s
			2617	Real-life	12 min
Baugh Jr *et al* (1998)	Cluster-first, route-second, simulated annealing	Min total distance, min number of vehicles, min inconvenience	25	Random	NA
			300	Real-life	NA
Cordeau and Laporte (2003)	Tabu search	Min transportation cost, min penalty violations	144	Random	93 min
			295	Real-life	768 min
Jih *et al* (2002)	Genetic algorithm	Min transportation cost, min penalty violation	100	Random	38 min

Note that Jih *et al* (2002) solves the related pickup-and-delivery problem.
NA = not applicable.

the computational experiments, the algorithm is tested on a number of simulated data sets with 250 customers, and four or five vehicles and real data sets with 2617 customers and 28 vehicles. The execution time for the simulated data set is about 20 s and about 12 min for the real data set.

In Baugh Jr *et al* (1998), the DARP is solved using simulated annealing. The work is based on the classical cluster-first, route-second approach. Customers are first organized into clusters and then the routes are developed for each individual cluster. The clustering is performed using simulated annealing while the routing is performed using a modified space–time nearest-neighbour heuristic. The modified space–time nearest-neighbour heuristic used to create routes for each cluster is a greedy algorithm. The results obtained are based on a set of real-life data set with 300 customers as well as on a generated data set with 25 customers. No CPU times are given in the paper. It is claimed by the authors that the algorithm gives near-optimal solutions.

Cordeau and Laporte (2003) describe a tabu search heuristic for the problem. Their algorithm initiates with a randomly generated initial solution. In order to avoid cycling, solutions possessing attributes of recently visited solutions are put on the tabu list and are therefore forbidden for a number of iterations. Infeasible solutions may be explored during the search, as constraints are relaxed and violations are added to the objective function. After each iteration, the parameters on the violations are dynamically adjusted. The objective function consists of the total transportation cost of the vehicles and the violation terms. Three variants of the heuristic were tested using both randomly generated data sets with 24–144 customers and six real-life data sets containing either 200 or 295 customers. The execution times for the randomly generated data sets are about 2 min for the smallest data sets and up to 93 min for the largest data sets. The execution times for the real-life data sets are given to be between 13 and 268 min.

Jih *et al* (2002) solve the single vehicle pickup and delivery problem with time windows using a GA. In the algorithm, the chromosome representation of a route is defined by letting the chromosome represent the locations in a travelling sequence of the route. The algorithm permits exploration of infeasible solutions during the search. The objective function is the sum of the total travel cost of the vehicle and the penalty for violating constraints. Four different types of crossover are considered. The algorithm is tested on randomly generated data sets with up to 100 customers. Execution time is about 38 min for the largest data sets. The results of the algorithm are compared with the optimal values, which are available for the smaller data sets (up to 40 customers). The best results for the GA are obtained by using the uniform order-based crossover. It is able to reach the optimum on the average in 83% of the runs for the data sets with up to 40 customers (Table 1).

4. The GA

GAs have shown good performance on a number of related routing problems (see eg Homberger and Gehring, 1999; Pereira *et al*, 2002). GA maintains a set of solutions, called a population. GA's are a part of evolutionary computing, inspired by Darwin's theory of evolution, that is, problems are solved by an evolutionary process resulting in a best (in

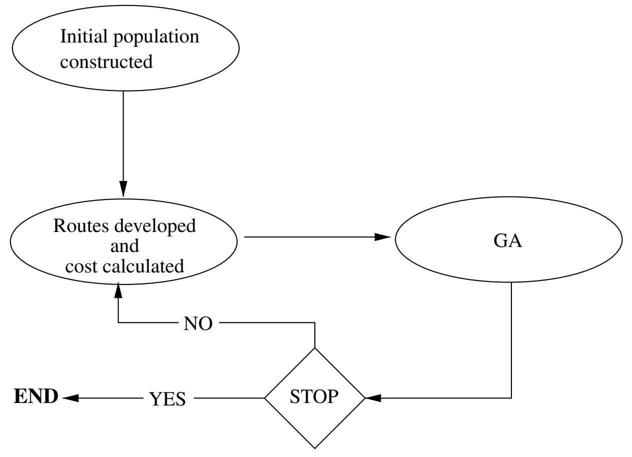
Initial population constructed
Routes developed and cost calculated
GA
NO
END
YES
STOP

Figure 3 An overview of the solution process.

GA-terms denoted the fittest) solution. The GA performs a number of iterations, in each of them, a number of individuals (solutions) are selected for reproduction. From these and the existing population, a new population is generated. For an in-depth introduction into GA, see Reeves (1995, pp 151–188). Therefore, an initial population generator is needed before the GA can start. In this case, the GA is initialized by creating an initial population in which all the customers are clustered randomly.

One way of implementing a GA for the DARP in one step is to adopt the chromosome representation used in Pereira *et al* (2002). In the chromosome representation, both the allocation of customers to vehicles and the order of the customers on the routes are encoded. The representation is used when solving the vehicle routing problem, but the extension to the DARP is problematic. The main obstacles are the precedence constraints (5) and (6). In order to solve this problem, some elaborate fix-up would be needed each time a new individual is created.

Instead, the classical split in a clustering phase and a routing phase is used. In the clustering part, as many groups of customers as there are available transportation vehicles are created. Each customer can only belong to one group and only one group can be assigned to each vehicle. The clustering of customers is solved using the GA. When all the customers have been grouped, a route for each vehicle is constructed. The routing involves deciding the order of the stops of the vehicle as well as the time table for the vehicle. The routing is solved using an extended version of the modified space–time nearest-neighbour heuristic developed by Baugh Jr *et al* (1998). Figure 3 presents an overview of the solution process.

The construction of the GA involves decisions regarding which chromosome representation fits the solution of the problem, which population size is adequate, how the initial population should be generated, what stopping criteria to use, how the fitness calculations should be made, what kind of selection mechanism is superior, and which modifying operators to use.

In order for an individual in the population to represent a solution to the problem of allocating customers to vehicles, it is decided to use a two-level binary chromosome representation. The representation is set up as a matrix. There are as many rows as there are available vehicles and there are as many columns as there are customers and depots. A 1-entry at any position $[g1, g2]$ indicates that the customer/depot represented by column $g1$ is allocated to the vehicle represented by row $g2$. It is very easy to verify that each customer is allocated to exactly one vehicle.

Figure 4 shows an example of what the chromosome representation looks like when there are four vehicles available, one depot and 16 customers. The routes can be constructed in many ways, for example, as shown in Figure 5.

The size of the population greatly influences the performance of the GA. If the population size is too small, it results in a high possibility of an under-covered solution space, while too large a population is a burden on the computational time and may lead to an unacceptable slow rate of convergence. It was decided to use the conventional method of a constant population size and perform some preliminary tests to decide how big the population size should be for the problems that are to be solved. The algorithm terminates after a fixed number of iterations.

In order to perform the crossover, we need to select two parents. One parent is chosen by means of a stochastic procedure, while the second parent is chosen randomly. The stochastic procedure, also called the roulette wheel method, gives each individual in the current population a probability of being chosen as parent proportional to the fitness value of the individual.

In each iteration, one offspring is created, which replaces one random member belonging to the Z portion of the current population, consisting of the individuals with the worst fitness values, that is, which represent solutions with high cost. Z is set to be proportional to the population size and different values for Z will be tested. This method for updating the population is called incremental replacement and the best member of a current population is guaranteed to survive to the next population.

The crossover used in the algorithm is a version of the crossover described by Pereira *et al* (2002). In this crossover, one row, that is, one cluster, is chosen at random from both parents and a random binary template is created. The template is used as a recipe for one row in the offspring, where a 1-entry in the template indicates that the gene is to be taken from parent 2. The offspring consists of the new row while the other rows are duplicates from parent 1.

When constructing a new solution with the crossover operator as described above, the resulting solution is not necessarily a legal solution. Therefore, it is examined whether a customer exists, which is assigned to more than one vehicle or no vehicle at all. If such a customer is found, a cluster is

	depot	customer 1	customer 2	customer 3	customer 4	customer 5	customer 6	customer 7	customer 8	customer 9	customer 10	customer 11	customer 12	customer 13	customer 14	customer 15	customer 16
Route 1	1	0	1	0	1	0	0	0	1	1	1	0	0	1	0	1	0
Route 2	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0
Route 3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Route 4	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1

Figure 4 An example of the binary chromosome representation used in the solution method. There are as many columns as customers and depots and as many rows as routes. The number of routes equals the number of available vehicles. Each customer must be assigned to exactly one route and each route has to include the depot.

depot - 9.1 - 15.1 - 15.2 - 2.1 - 9.2 - 8.1 - 10.1 - 8.2 - 2.2 - 10.2 - 13.1 - 13.2 - 4.1 - 4.2 - depot

Figure 5 A possible route based on the clustering of Figure 4, where the $i.1$ is the pickup location of customer i and $i.2$ is the drop off location of customer i, for all i in route 1.

chosen randomly and the customer is either added or deleted from that cluster depending on whether the customer is allocated too often or not at all. If the randomly chosen cluster is the cluster created in the crossover, a new cluster is randomly chosen. This procedure is repeated until all the customers are allocated to exactly one vehicle.

Initially, the template was generated completely randomly, but tests indicate (more information on the tests can be found in Bergvinsdottir (2004)) that increasing the probability of selecting genes from parent 1 yields better results. Preliminary results show that a big increase of the probability is not sensible, especially in the beginning of the iterations, therefore the probability of choosing genes from parent 1 (the better parent) is set to 60%.

The mutation operator moves one random customer from its current cluster to another random cluster. It is not possible to generate illegal solutions using this mutation so no verification or correction procedure is needed after mutation. The offspring that has been created in the crossover can be subjected to mutation with a certain probability, called the mutation probability, P_μ.

As duplicates distort the selective process and waste computational resources, they are to be avoided if possible. In our algorithm, the probability of duplicate chromosomes is reduced by mutating the offsprings that have the same fitness values as their parent 1 and therefore represent the same solution.

Now clusters of customers have been generated using the GA in the first phase of the solution method. In the second phase, routes are constructed for the clusters and costs evaluated. The modified space–time nearest-neighbour heuristic (Baugh routing heuristic) described by Baugh Jr *et al* (1998) is used, in an extended version, for developing routes and calculating cost for each route. It is used as it displays excellent results.

The Baugh routing heuristic is a greedy heuristic based on the space–time nearest-neighbour heuristic. The aim of the heuristic in our GA is to construct a good route given a set of customers with pickup and delivery locations along with time windows for both locations.

The heuristic starts by selecting the first customer in the route. The first customer is the customer with the earliest pickup time. The pickup location of the first customer is the first stop in the route. To find the next stop, four candidate stops are considered. The candidate stops are the four stops that are closest, in space and time, to the first customer's pickup location. The next stop is chosen as the cheapest stop of the four candidate stops. The cost of a stop is evaluated as the cost of the possible next three succeeding moves after that stop. This procedure is repeated until a route consisting of all the customers pickup and drop off locations has been generated.

The selection of the four stops, which are taken into consideration as the next stop in the route, is performed as follows. First, the drop off locations of the customers already in the vehicle are considered and the one closest to the current stop in space and time is chosen. If there is an empty seat in the vehicle, then the pickup locations of customers that have not yet been serviced are considered. If there exists such a pickup location that is closer to the current stop in space and time than the closest drop off location (if any), then that stop is chosen as the first of the four stops, otherwise the drop off location is set as the first. This procedure is repeated three times, which results in four possible next stops.

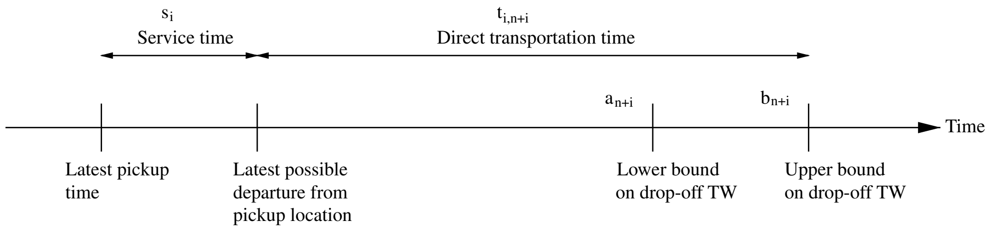
s_i
Service time
$t_{i,n+i}$
Direct transportation time
a_{n+i}
b_{n+i}
Time
Latest pickup time
Latest possible departure from pickup location
Lower bound on drop-off TW
Upper bound on drop-off TW

Figure 6 Illustration of the latest pickup time calculations for an outbound customer. The latest pickup time is calculated backwards in time. The direct transportation time from pickup to drop-off location and the service time at the pickup location are subtracted from the upper bound on the drop-off time window.

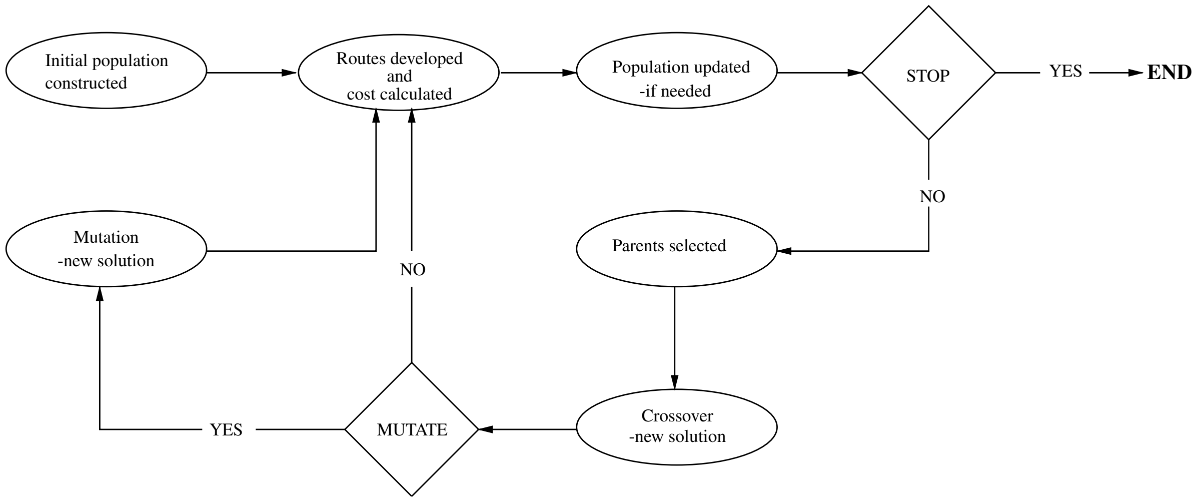
Initial population constructed
Routes developed and cost calculated
Population updated -if needed
STOP
YES
END
NO
Parents selected
Mutation -new solution
NO
Crossover -new solution
MUTATE
YES

Figure 7 An overview of the initial heuristic.

The closeness of two stops is measured using a space–time separation between the stops. The space–time separation between two stops is quantified by a weighted sum of travel time between the stops and the time window violation at the latter stop. The time window violation is positive if the latest time at which a destination stop can be reached precedes its time window. The time window violation is negative if the earliest time at which a destination stop can be visited exceeds its time window.

The cost of a move between two locations is set to be the weighted sum of travel time and the absolute amount by which the time window is violated at the latter location.

An illustration of the latest pickup time calculations for an outbound customer is shown in Figure 6. Figure 7 gives an overview of the structure of the heuristic.

5. Experimental results

There are as yet no well-known and established benchmarks available in the literature for the version of DARP used in this project. The behaviour of the solution method proposed here can therefore not be tested using data instances that have been widely tested and the results cannot be compared to the optimum or the best results obtained in previous publications.

The test instances used for testing the solution method proposed here are obtained from Cordeau and Laporte (2003). They created the test instances to analyse the behaviour of a simulated annealing algorithm for solving the DARP. Cordeau and Laporte generated a set of 20 random test instances (data sets) according to realistic assumptions. The information regarding time window widths, vehicle capacity, route duration, and maximum ride time were provided by the Montreal Transit Commission (MTC).

In the test instances, there are between 24 and 144 customers. The first half of the customers is assumed to consist of outbound customers while the remaining customers are assumed to be inbound. For each instance, the origin and destination locations are generated using a procedure that creates clusters of vertices around a certain number of seed points. All instances contain a single depot and the location of the depot is set at the average location of the seed points. For a more detailed description of this procedure, we refer to Cordeau *et al* (1997).

For each instance, the service time (s_i) in each location i ($i \in N$) is equal to 10, and the load change, l_i, in each location

Table 2 Size of data instances used in tests

Instances		*Customers*	*Vehicles*
R1a	R1b	24	3
R2a	R2b	48	5
R3a		72	7
R4b		96	9
R5a	R5b	120	11
R6a	R6b	144	13
R07a	R07b	36	4
R9a	R9b	108	8
R10a	R10b	144	10

i is either 1 or -1. 1 for the pickup locations and -1 for the drop-off locations, that is, no customer has companions travelling with them, and all the customers demand only one seat. The depot location, on the other hand, has a service time and load change equal to zero, since no customers are entering or leaving the vehicle at the depot. Further, the maximum route duration, r^k, is equal to 480 in all instances, vehicle capacity, C^k, is equal to 6, and the maximum ride time, u_i, is equal to 90.

A time window $[a_i, b_i]$ is generated for each location. As mentioned in Section 2, the origin point of an inbound customer and the destination point of an outbound customer are subject to time windows, while the other points have no customer-specific time windows associated with them. The time windows for these points are therefore set to $[0, T]$, where T is the planing horizon, which in these experiments is equal to 1440, that is, the number of minutes in one day.

Two groups of customer-specific time windows are constructed in the data sets. The first one has narrow time windows, while the second one has wide time windows. The narrow time windows are constructed by choosing an uniform random number, a_i, in the interval [60,480] and then choosing another uniform random number, b_i, in the interval $[a_i + 15, a_i + 45]$. For the wide time windows group, a_i is chosen in the same manner but b_i is chosen in the interval $[a_i + 30, a_i + 90]$, resulting in time windows $[a_i, b_i]$ $(i \in N)$. The test instances R1a to R10a have narrow time windows, while test instances R1b to R10b have wide time windows.

Test instances R1a to R6a and R1b to R6b are generated so that the number of available vehicles in comparison to the number of customers is higher than in test instances R7a to R10a and R7b to R10b, for example, R6a has 13 available vehicles while R10a has 10 available vehicles, both instances having 144 customers.

All the test instances constructed by Cordeau and Laporte are available from the Internet at http://www.hec.ca/chairedistributique/data/darp/ (accessed 11th January 2006). We are only considering the instances where the necessary information for a comparison is present. Problems in Table 2 have a given solution, except R4b and R7a, which will be used in parameter tuning.

Table 3 Fixed parameters

Population size	M	50
Iteration number	G	15 000
Mutation probability	P_μ	0.01
Proportion replaced	Z	0.10

Table 2 gives the number of customers and available vehicles in the test instances that are used in this paper.

The distance between any two locations i and j is set to be the Euclidean distance between the coordinates of locations i and j, $i, j \in A$. The speed of the vehicles is set to 1, so the transportation time $t_{i,j}$ is equal to the Euclidean distance between i and j. So the first term in the objective function 1 now equals the total weighted transportation distance.

Through an extensive set of tests, good values for the population size, number of iterations, mutation probability, and proportion of population that is replaced have been found. The parameters are fixed at the values reported in Table 3. A more in-depth description of the tests can be found in Bergvinsdottir (2004).

Of course, the more iterations that are run, the better solutions are obtained. The chosen number of iterations defines a good compromise between time and quality. Running, for example, 30 000 iterations resulted in execution times of around 150 min, which is too much for a practical setting.

The customers naturally emphasize the level of service provided by the transportation operator. They want the bus to arrive on time and the ride time to be minimal. On the other hand, they do not want the service to be very expensive and therefore they set their demands on service to reasonably high levels in their opinion. After performing some preliminary testing, the resulting weights are:

$$w_1 = 8,\ w_2 = 3,\ w_3 = 1,\ w_4 = 1,\ w_5 = n,\ w_6 = n,\ w_7 = n$$

The weights on violating the relaxed constraints presented in the objective function (1) are set to n, that is, the total number of customers in each data instance, because the values for the cost factors for distance, route duration, and ride time increase proportionally with the number of customers. The value of the cost terms for the relaxed constraints is not as dependent on the number of customers as the above-mentioned factors. The weight on customers transportation time is set to 8, as the customers are concerned with the transportation time. The weight on excess ride time (w_2) is set to 3, as the customers consider it important that transportation time is short. The weights on waiting time with customers (w_3) are set to 1, because the customers generally do not mind waiting in the vehicle as long as the excess ride time is reasonable. The weight on work time (w_4) is also set to 1, because the size of the route duration is large compared to the other segments in the fitness function and the customers are not very concerned with the route duration, as they only share part of the route.

Table 4 The results obtained by the genetic algorithm

	Route duration		*Vehicle waiting time*				*Ride time*				*CPU time (min)*
			Avg.		*Best*		*Avg.*		*Best*		
	Avg.	*Best*	*Total*	*Avg.*	*Total*	*Avg.*	*Total*	*Avg.*	*Total*	*Avg.*	
R1a	1041	1039	252	5.25	260	5.42	477	19.86	310	12.90	5.57
R2a	1969	1994	470	4.90	514	5.36	1367	28.47	1330	27.72	11.43
R3a	2779	2781	292	2.03	301	2.09	3081	42.79	2894	40.20	21.58
R5a	4250	4274	500	2.08	527	2.20	5099	42.49	4837	40.30	58.23
R9a	3597	3526	94	0.44	32	0.15	6251	57.88	6719	62.21	40.78
R10a	5006	5025	315	1.09	246	0.86	8413	58.42	8341	57.92	65.98
R1b	907	928	143	2.98	164	3.42	630	26.24	549	22.89	5.46
R2b	1719	1710	198	2.06	162	1.69	1214	25.30	1300	27.07	11.72
R5b	4296	4336	552	2.30	568	2.37	4615	38.46	4720	39.33	58.93
R6b	5309	5227	630	2.19	513	1.78	6134	42.59	6397	44.42	81.23
R7b	1299	1316	102	1.41	128	1.78	990	27.50	784	21.76	8.29
R9b	3679	3676	147	0.68	177	0.82	5362	49.65	5358	49.61	44.66
R10b	4733	4678	113	0.39	85	0.29	7969	55.34	8119	56.38	66.41
Total	40 584	40 508	3808	27.81	3678	28.21	51 600	514.99	51 657	502.72	488.61

Table 5 Results obtained by Cordeau and Laporte (2003)

	Route duration	*Vehicle wait. time*		*Ride time*		*CPU (min)*
		Total	*Avg.*	*Total*	*Avg.*	
R1a	881	211	4.4	1095	45.62	1.90
R2a	1985	724	7.54	1977	41.18	8.06
R3a	2579	607	4.22	3587	49.82	17.18
R5a	3870	833	3.47	6154	51.3	46.24
R9a	3155	323	1.5	5622	52.05	50.51
R10a	4480	721	2.5	7164	49.75	87.53
R1b	965	321	6.68	1042	43.4	1.93
R2b	1565	309	3.22	2393	49.86	8.29
R5b	3596	606	2.52	6105	50.87	54.33
R6b	4072	449	1.56	7347	51.02	73.70
R7b	1097	129	1.79	1762	48.94	4.23
R9b	3249	487	2.26	5581	51.68	51.28
R10b	4041	362	1.26	7072	49.11	92.41
Total	35 537	6082	42.92	56 900	634.6	497.59

In all tests, each instance is run five times and we present the average results from the data obtained in these runs along with the best total cost found for each instance.

The results obtained using the GA is compared to the results obtained by Cordeau and Laporte (2003). The average results from the best improved heuristic are presented in Table 4 along with the results for the best solution obtained for each data set (ie the solution with the lowest cost). The vehicle waiting time is not critical in this model, since it is not a part of the objective to minimize the vehicle waiting time but rather customer waiting time. Cordeau and Laporte (2003), on the other hand, report this waiting time which is the reason it is included in Table 4. The best solution results are somewhat different from the average results of Cordeau and Laporte (2003). Route duration is 14% higher, waiting time 40% lower, and ride time 9% lower in the solution presented here.

Cordeau and Laporte (2003) performed 10^4–10^5 iterations in order to get their best results, which are presented in Table 5. In the testing in this paper, 15 000 iterations are performed. The total CPU time is however comparable. Cordeau and Laporte (2003) use an Intel Pentium 4, 2 GHz processor in their CPU measurements, and in this paper, results are obtained on an Intel Celeron 2 GHz processor. Ninety-nine per cent of the CPU time spent on the test cases in this paper is used by the routing heuristic.

The objective function used in Cordeau and Laporte (2003) is

$$v(S) = v_1(S) + \alpha v_2(S) + \beta v_3(S) + \gamma v_4(S) + \tau v_5(S) \quad (16)$$

where $v(S)$ is the objective value for solution S. The function $v_1(S)$ denotes the total routing cost of the vehicles, $v_2(S)$ denotes total load violation, $v_3(S)$ denotes total route duration violation, $v_4(S)$ denotes total time window violation, and $v_5(S)$ denotes total ride time violation. α, β, γ, and τ are a self-adjusting weights. The values of these weights change in each iteration and no approximate values are given in the paper. Neither is the method used to calculate the total routing cost specified in the paper, which makes it impossible to evaluate the comparison of the total costs obtained by the two solution methods.

As mentioned previously, the route duration is higher in our algorithm than in Cordeau and Laporte (2003). The other two results, which are related to customer service, ride time, and vehicle waiting time are, on the other hand, better than in the results obtained by Cordeau and Laporte (2003). One reason for these results is that, in this paper, the weights are set to represent the choice of the customers so there is an emphasis on customer service factors.

6. Conclusion and further research

This paper has presented an implementation of a new heuristic approach for solving the DARP, using the classical cluster-first, route-second framework. A GA was used for the clustering phase, and a modified space–time nearest-neighbour heuristic was used in the routing part.

The resulting method was compared to the results given by Cordeau and Laporte (2003). The comparison focused on route duration, as well as ride and vehicle waiting times. The comparison showed that Cordeau and Laporte (2003) obtain better results for route duration, whereas the GA approach presented here obtains better results with regards to ride time and vehicle waiting time. The results are overall comparable to those obtained by Cordeau and Laporte (2003), where differences are explained by the setting of cost *versus* service level parameters.

The improvements to the GA introduced in Section 4 regarding the reduction of randomness have shown to give very good results, and several ideas concerning further improvements are still untested. One possibility is to use ranking when selecting parents in the GA and to experiment with different crossover and mutation operators. Also, to improve computational time, a different routing algorithm could be implemented. The modified space–time nearest-neighbour heuristic used 99% of the CPU time, which of course set a rather low limit on the number of iterations the GA could perform.

The overall conclusion of this paper is that the new solution method for solving the DARP, which is presented here, shows some promising results. It is possible to adjust the weights of seven factors concerning cost of operation *versus* level of service, which enables evaluation of consequences in different Dial-a-Ride scenarios. The new solution method has achieved solutions comparable to the current state-of-the-art methods.

References

Baugh Jr JW, Kakivaya DKR and Stone JR (1998). Intractability of the Dial-a-Ride problem and a multiobjective solution using simulated annealing. *Eng Optim* **30**: 91–124.

Bergvinsdottir KB (2004). *The genetic algorithm for solving the Dial-a-Ride problem*. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark.

Borndorfer R, Grötschel M, Klostermeier F and Kuttner C (1997). Telebus Berlin: Vehicle scheduling in a dial-a-ride system. Technical report, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Cordeau JF, Gendrau M and Laporte G (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **30**: 105–119.

Cordeau JF and Laporte G (2003). A tabu search heuristic for the static multi-vehicle Dial-a-Ride problem. *Transport Res B-Meth* **37**: 579–594.

Homberger J and Gehring H (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR* **37**: 297–318.

Jaw JJ, Odoni AR, Psaraftis HN and Wilson NHM (1986). A heuristic algorithm for the multi-vehicle advance request Dial-a-Ride problem with time windows. *Transport Res B-Meth* **20B**(3): 243–257.

Jih WR, Kao CY and Hsu FYJ (2002). Using family competition genetic algorithm in pickup and delivery problem with time window constraints. In: *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*. IEEE, Piscataway, NJ, pp 496–501.

Jorgensen RM (2002). *Dial-a-Ride*. PhD thesis, Center for Traffic and Transportation, Technical University of Denmark.

Pereira FB, Tavares J, Machado P and Costa E (2002). Gvr: A New genetic representation for the vehicle routing problem. In: *Proceedings of Artificial Intelligence and Cognitive Science. 13th Irish Conference, AICS 2002, volume 2464 of Lecture Notes in Artificial Intelligence*. QAD Ireland, Limerick, Ireland, pp 95–102.

Reeves CR (ed). *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill: London.

Received November 2004;
accepted June 2006 after two revisions