4OR (2007) 5:61–73
DOI 10.1007/s10288-006-0018-0

REGULAR PAPER

An effective and fast heuristic for the Dial-a-Ride problem

Roberto Wolfler Calvo · Alberto Colorni

Received: 15 June 2005 / Accepted: 15 June 2006 /
Published online: 26 July 2006
© Springer-Verlag 2006

Abstract Dial-a-Ride is an emerging transport system, in which a fleet of vehicles, without fixed routes and schedules, carries people from the desired pickup point to the desired delivery point, during a pre-specified time interval. It can be modeled as an $\mathcal{NP}$-hard routing and scheduling problem, with a suitable mixed integer programming formulation. Exact approaches to this problem are too limited to tackle real-life instances (hundred of trips), therefore heuristics are needed. The heuristic method proposed in this paper builds an auxiliary graph and then solves an assignment problem on this graph. The auxiliary graph is obtained by replacing pairs of nodes with a single one and associating an ad hoc cost function to the new set of arcs. Two different simple methods are employed to transform the infeasible solution given by the assignment problem into a feasible one. The proposed algorithms have been tested on instances created using the Milan network and shown to be fast and effective.

Keywords Dial-a-Ride · Routing · Scheduling · Heuristic

MSC classification 90B06 · 90C11 · 90C59

R. Wolfler Calvo (✉)
Institut Charles Delaunay (ICD), FRE CNRS 2848,
University of Technology of Troyes,
BP 2060, 10010 Troyes, France
e-mail: roberto.wolfler@utt.fr

A. Colorni
Departement of Industrial Design,
Arts and Comunications (INDACO), Politecnico di Milano,
Piazza Leonardo da Vinci, 32, 20133 Milano, Italy
e-mail: alberto.colorni@polimi.it

1 Introduction

The Dial-a-Ride (DAR) system concerns the management of a fleet of vehicles used for public transport. The customers demand the service by calling a central unit and specifying: the desired pickup point, the delivery point (respectively, *origin* and *destination*), the number of passengers and some limitations on the service time (e.g., the earliest departure time). Such transportation system is called *demand-responsive*: the routes and schedules of the vehicles change dynamically on the basis of the current requests of the users. By better exploiting vehicle capacity, they offer the comfort and flexibility of private cars and taxies at a lower cost. DAR is suited to service sparsely populated areas, or densely populated areas during weak demand periods or for special classes of passengers with specific requirements (elderly, disabled).

Several models of DAR service have been proposed in the literature: with or without time windows, with a fixed or unlimited fleet of vehicles and so on. In the "static" DAR, the customers ask for service in advance and the plan is made before the system starts to operate; in the "dynamic" DAR, the customers can call during the service period (see Savelsbergh and Sol 1995) and the solution is updated on-line. Different objective functions have been taken into account: minimization of the number of vehicles used or of the total travel time, maximization of the number of customers served or of the level of service provided to the user. This paper addresses the static DAR Problem (DARP) with time windows and a fleet of fixed size. Each request corresponds to a single passenger, and the objective function maximizes firstly the number of customers served, and then the level of service provided on average.

DARP is $\mathcal{NP}$*-hard* in the strong sense, as it generalizes the *Pickup and Delivery Problem with Time Windows* (PDPTW). Moreover, it is important to stress that the number of vehicles is fixed and that even finding a feasible solution is itself a $\mathcal{NP}$-complete problem (see, e.g., Savelsbergh 1985). Nevertheless, exact algorithms for the single-vehicle case have been developed by Psaraftis (1980, 1983), Desrosiers et al. (1986); Sexton and Choi (1986). The last reference, in particular, considers the minimization of customer uncomfort. An exact approach for the multi-vehicle case has been proposed by Dumas et al. (1991). Recently, a Branch and Cut algorithm for this problem has been proposed by Cordeau (2003). However, most of the time the exact approaches are unable to solve real-life instances (typically hundreds of trips).

As for heuristics, parallel insertion algorithms have been presented by Jaw et al. (1986) and Madsen et al. (1995). A heuristic for the transport of handicapped people with a homogeneous fleet of vehicles has been proposed by Ioachim et al. (1995). In Toth and Vigo (1997) a specific heuristic for a particular type of DARP related to the transport of handicapped people with an heterogeneous fleet of vehicles has been presented. Finally, Cordeau and Laporte (2003b) described a tabu search heuristic for a variant of the problem addressed in this paper. For an overview on PDPTW, DARP and related problems,see Dumas et al. (1991), Savelsbergh and Sol (1995) and Cordeau and Laporte (2003a).

This paper describes a very fast, simple and effective heuristic to solve the DARP. The main contribution of the paper is the adaptation of a method similar to the one proposed in Wolfler Calvo (2000) to a more complex problem. The algorithm builds an auxiliary graph having one node (instead of two) for each customer. Each new arc connecting the new nodes has an associated cost function measuring both space and time distance between two customers (in two different ways). This graph can be obtained thanks to several non-trivial transformations on the original formulation (i.e., several families of constraints are relaxed and used to build the cost function). The number of vehicles being fixed to *nv*, the solution given by the assignment problem defined on the auxiliary graph is a set of *nv main paths* (one for each vehicle) and a very small set of subtours. A main path is a chain of nodes (customers) containing the node depot. The subtours involve only customers (two or three on average, hardly more), thus are easily inserted in the main paths. The chains of customers represent a first step towards a feasible solution for the original problem, since all they are missing is the set of delivery nodes. The algorithm inserts the delivery nodes both in a straightforward way (each delivery node follows the correspondent pickup node) and in a less trivial one. The next section presents the mathematical formulation. The method is illustrated in Sect. 3. Computational results and conclusions close the paper.

2 The problem

Let $R = \{1, \ldots, n\}$ be a set of requests (customers). For each request i, two nodes (i^+ and i^-) are defined: a load q_i must be taken from i^+ and delivered to i^-. Let $N^+ = \left\{i^+ | i \in R\right\}$ be the set of pick up nodes and $N^- = \left\{i^- | i \in R\right\}$ denote the set of delivery nodes. A positive amount $q_{i^+} = q_i$ is associated to the pickup node, a negative amount $q_{i^-} = -q_i$ to the delivery node. A time window is also associated with each node, whether it is a pickup node $[e_{i^+}, l_{i^+}]$ or a delivery node $[e_{i^-}, l_{i^-}]$. The fleet of vehicles is denoted as V; all vehicles have the same capacity Q and time window at the depot $\left[e_0, l_0\right]$. Let $G = (N, A)$ be a directed graph, whose set of vertices is defined as $N = N^+ \cup N^- \cup \{0\}$, where node 0 represents the vehicle depot. The set of arcs is defined as $A = \{(i,j) : i,j \in N, i \neq j\}$ and each arc $(i,j) \in A$ has an associated distance $d_{i,j}$ or a travel time $t_{i,j}$. Another set $E = \left\{(i,j) : i,j \in N^+ \cup N^-, i \neq j\right\}$ represents the subset of arcs whose adjacent nodes are customer nodes.

The problem is to find a set of routes starting and ending at the depot, such that all the customer requests are satisfied and the pickup node of each customer is visited before the delivery node. Moreover, the solution should be feasible with respect to the capacity constraints and the time window constraints. The variables $x^v_{i,j}$ are equal to 1 if vehicle v uses arc $(i,j) \in A$ and equal to 0 otherwise; p_i represents the departure time from node $i \in N^+ \cup N^-$; y_i is the load of the vehicle leaving node i. The problem is defined as follows:

$$\max \sum_{i \in N^+} \alpha_i \sum_{v \in V} \sum_{j \in N} x^v_{i,j} - \sum_{i \in N^-} S_i \tag{1}$$

s.t.

$$\sum_{v \in V} \sum_{j \in N} x^v_{i,j} \leq 1 \quad \forall i \in N^+, \tag{2}$$

$$\sum_{j \in N} x^v_{i,j} - \sum_{j \in N} x^v_{j,i} = 0 \quad \forall v \in V, \ \forall i \in N^+ \cup N^-, \tag{3}$$

$$\sum_{j \in N} x^v_{i^+,j} - \sum_{j \in N} x^v_{i^-,j} = 0 \quad \forall v \in V, \ \forall i^+, i^- \in N^+ \cup N^-, \tag{4}$$

$$x^v_{i,j}(y_i + q_j) \leq y_j \quad \forall v \in V, \ \forall (i,j) \in E, \tag{5}$$

$$q_i \leq y_i \leq Q \quad \forall i \in N^+, \tag{6}$$

$$0 \leq y_i \leq Q - q_i \quad \forall i \in N^-, \tag{7}$$

$$x^v_{i,j}(p_i + t_{i,j}) \leq p_j \quad \forall v \in V, \quad \forall (i,j) \in E, \tag{8}$$

$$e_i \leq p_i \leq l_i \quad \forall i \in N, \tag{9}$$

$$p_{i^+} + t_{i^+,i^-} \leq p_{i^-} \quad \forall i = i^+, i^- \in N^+ \cup N^-, \tag{10}$$

$$x^v_{i,j} \in \{0,1\} \quad \forall v \in V, \ \forall (i,j) \in A. \tag{11}$$

The objective function (1) has two terms. The first term is the number of serviced customers, suitable weighted with the coefficient α. The second term represents the level of service over the set of serviced customers. A reasonable measure of the quality of service perceived by the customers (indicated in the following with S) is introduced as the ratio between the service time offered by the DAR system for the trip and the minimum time the customer would need to go from the origin to the destination (i.e., the value associated to the arc (i^+, i^-)). More precisely, the quality of service for customer i is defined as

$$S_i = \frac{(a_{i^-} - e_{i^+})}{t_{i^+ i^-}}, \tag{12}$$

where a_{i^-} is the arrival time in node i^-, $(a_{i^-} - e_{i^+})$ denotes the total time needed to reach the destination using the DAR transportation system and $t_{i^+ i^-}$ denotes the minimum time needed to go from the origin to the destination directly. Obviously $S_i \geq 1 \ \forall i$ and the higher its value, the lower the service quality for customer i. Quantity α is high enough to guarantee that maximizing the number of serviced customers is the main objective.

The first three groups of constraints ensure that each customer is serviced by at most one vehicle. More precisely, constraints (2) make sure that at most one vehicle exits from each origin node i^+. Constraints (3) impose that the number of vehicles entering and exiting each node has to be the same. Finally, constraints

(4) establish that the same vehicle, if any, visits the pickup and the delivery node (pairing constraints). Constraints (5), (6) and (7) ensure the feasibility of the loads: (a) the number of passengers in a given vehicle varies according to the number of people boarding it or getting off it; (b) the maximum capacity of the vehicle cannot be exceeded. The last three classes of constraints impose the feasibility of the schedule. Constraints (8) represent the compatibility requirements between routes and schedules. Constraints (9) ensure that the departure takes place during the time window: when the vehicle arrives at node i before time e_i the driver must wait; it is infeasible to arrive at node i after l_i. Finally, constraints (10) imply that for each trip the delivery node is visited after the pickup node. Constraints (8) and constraints (5) can be linearized respectively as $M(1-x^v_{ij}) \geq p_i + t_{ij} - p_j$ and $P(1-x^v_{ij}) \geq y_i + q_j - y_j$. The former equations are a generalization of the classical *TSP subtour* elimination constraints proposed by Miller et al. (1960).

3 The algorithm

The framework of the proposed heuristic can be described as follows:

step 1 Build the auxiliary graph (which turns out to be smaller than the original one).
step 2 Calculate the fleet dimension (nv).
step 3 Add the nv nodes to the graph.
step 4 Solve the assignment problem.
step 5 Insert the subtours in the main paths.
step 6 Modify the infeasible paths so that they become feasible.

The auxiliary graph has one node for each customer and no associated time window. For each pair of nodes an arc (i, j) is defined with associated cost $\overline{p}_{ij}$. This cost coefficient measures the proximity of two customers i, j by combining their spatial and temporal distances. These two distances are obtained using the information contained in the subgraph induced by $\{i^+, i^-, j^+, j^-\}$ (i.e., travel time, waiting time and time windows). The algorithm then calculates a lower bound of the fleet size ($nv = |V|$) and adds another set of nv nodes to the graph. An assignment problem is then solved on the auxiliary graph.

The assignment problem is the easiest and classical relaxation for routing problems, but it is very weak since the solution obtained is hardly feasible in the original problem. The solution can be a long way from satisfying precedence constraints, time window constraints and pairing constraints. The assignment problem solved on the auxiliary graph, however, gives a solution which is close to a feasible one. Two procedures, transforming the infeasible solution in a feasible one, are the last steps of the algorithm.

A similar approach can be found in Wolfler Calvo (2000). Nevertheless, the TSPTW formulation is much easier to transform, since it is enough to relax the time windows constraints (i.e., the subtour elimination constraints). On the other hand, the DARP has a more complex formulation, since each customer is represented with two nodes imposing to introduce the precedence constraints (10), the pairing constraints (4) and the capacity constraints (5), (6) and (7). The following paragraph explains in detail how to reduce the problem in the DARP case.

3.1 The auxiliary graph

Denote the (directed) auxiliary graph by $\tilde{G} = (R, \tilde{A})$. The set of vertices is the set of customers R and the set of arcs is defined as $\tilde{A} = \{(i,j) : i,j \in R, i \neq j\}$. This graph has half of the nodes and less than a quarter of the arcs of G and there are no time windows. The coefficient $\overline{p}_{ij}$ associated to each arc of $\tilde{A}$ measures the distance between customer i and customer j by combining all the values (i.e., travel times, waiting times and time windows) necessary to evaluate the feasibility and the cost of any possible direct path from customer i to customer j, starting with node i^+ (i.e., $\{i^+, j^+, i^-, j^-\}$, $\{i^+, i^-, j^+, j^-\}$, $\{i^+, j^+, j^-, i^-\}$).

To compute $\overline{p}_{ij}$ constraints (8) and (9) are used. These constraints state that $p_j = \max(p_i + t_{ij}, e_j)$. When the sequence is composed by only two nodes ($x_{ij} = 1$), the equation becomes

$$p_j = \max(e_i + t_{ij}, e_j) = e_i + t_{ij} + \widetilde{w}_{ij}, \tag{13}$$

where $\widetilde{w}_{ij} = \max(e_j - e_i - t_{ij}, 0)$. For a generic sequence of s nodes $\Pi = \{\pi_1, \ldots, \pi_s\}$, the departure time in node s is defined as

$$p_{\pi_s} = p_{\pi_1} + T_{\pi_1,\pi_s} + W_{\pi_1,\pi_s} \tag{14}$$

where T_{π_1,π_s} is the total travel time along Π ($\sum_{k=1}^{s-1} t_{\pi_k,\pi_{k+1}}$), W_{π_1,π_s} is the total waiting time ($\sum_{k=2}^{s} w_{\pi_k}$, where $w_{\pi_k} = p_{\pi_k} - a_{\pi_k}$) and a_{π_1} and p_{π_1} are, respectively, the arrival and the departure time in the first node of the sequence (see Cordone and Wolfler Calvo 1996).

Applying Eq. (14) to the first sequence (i.e., $\{i^+, j^+, i^-, j^-\}$) it is possible to calculate $p^1_{j^-}$ as follows:

$$p^1_{j^-} = e_{i^+} + t_{i^+,j^+} + t_{j^+,i^-} + t_{i^-,j^-} + w_{j^+},$$

since, $\pi_s = j^-$, $\pi_1 = i^+$, $p_{\pi_1} = e_{i^+}$, $W_{\pi_1,\pi_s} = w_{j^+}$ and $T_{\pi_1,\pi_s} = T_{i^+,j^-} = t_{i^+,j^+} + t_{j^+,i^-} + t_{i^-,j^-}$.

There is only one possible waiting time, i.e., w_{j^+} in node j^+, since the sequence starts in i^+ and delivery nodes have no waiting time (i.e., $e_{i^-} = e_{i^+} + t_{i^+i^-}$). If $e_{i^-} > e_{i^+} + t_{i^+i^-}$, the customer should wait before getting off, which makes no sense; if $e_{i^-} < e_{i^+} + t_{i^+i^-}$, the time window can be reduced, since part of it will be never used (Desrosiers et al. 1995). So $p_{i^-} = a_{i^-}$.

In the same way it is easy to calculate $p^2_{j^-}$ for the second sequence: $p^2_{j^-} = e_{i^+} + t_{i^+,i^-} + t_{i^-,j^+} + t_{j^+,j^-} + w_{j^+}$, and the last one $p^3_{i^-} = e_{i^+} + t_{i^+,j^+} + t_{j^+,j^-} + t_{j^-,i^-} + w_{j^+}$. The values $\overline{p}_{ij}$ are defined as follows:

$$\overline{p}_{ij} = \frac{\sum_{r=1}^{3} p^r_{n(r)} y_r}{\sum_{r=1}^{3} y_r},$$

where $n = \{j^-, j^-, i^-\}$ and $y_r = 1$ if $p^r < \infty$ and $y_r = 0$ otherwise. Of course, when $\sum_{r=1}^{3} y_r = 0$ the procedure set directly $\overline{p}_{ij} = \infty$: there is no feasible way to go from customer *i* to customer *j*. The values $\overline{p}_{ji}$ are defined in the same way and it is clear that $\overline{p}_{ij} \neq \overline{p}_{ji}$. The value $\overline{p}_{ij}$ can be seen as the sum of two terms: one measures the straightforward path length (the travel time) and the other one measures the distance between the time windows (w_{j^+}). Notice that Toth and Vigo (1997) use an average travel time measure which is different from the one proposed here.

3.2 The fleet of vehicles

In order for the solution of the assignment problem to be useful, it would be desirable to enlarge the auxiliary graph with the minimum number of vehicles necessary to satisfy the requests. Since this number is unknown we settle for a lower bound. Two lower bounds for this number are proposed, namely b_1 and b_2. The fleet dimension nv is then set equal to the maximum of this two (i.e., $nv = \max(b_1, b_2)$).

b_1 is derived from the maximal set of customers which cannot be loaded by the same vehicle, since going from customer *i* to customer *j* violates some time windows. Let $G' = (R, A')$ denote the incompatibility graph derived from $\tilde{G} = (R, \tilde{A})$ linking each pair of incompatible customers (i.e., $\overline{p} = \infty$). On this new graph the algorithm looks for the maximal clique and b_1 is the cardinality of the maximal clique. The algorithm used is the one proposed in Carraghan and Pardalos (1990).

b_2 considers the time horizon T and it has been proposed in Kontoravdis and Bard (1995) for the Vehicle Routing Problem with Time Windows. Request *i* takes up an amount of time $\tilde{t}_i$ given by the travel time t_{ij} used to reach the next node along the route, and, possibly, a waiting time $w_j = \max(e_j - p_i - t_{ij}, 0)$. Hence, $\tilde{t}_i$ can be underestimated as $\tilde{t}_i = \min_{j \in N} \left[\max\left(t_{ij}, e_j - l_i - t_{ij}\right)\right]$ and all of the requests must be serviced from e_0 to l_0:

$$b_2 = \left\lceil \frac{\sum_{i \in N} \tilde{t}_i}{l_0 - e_0} \right\rceil. \tag{15}$$

3.3 The assignment problem

The assignment problem is therefore defined on a new complete graph $\hat{G} = (\hat{R}, \hat{A})$, as follows:

$$\min \sum_{(ij)\in \hat{A}} \overline{p}_{ij} x_{ij} \tag{16}$$

s.t

$$\sum_{j\in \hat{R}} x_{ij} = 1 \quad \forall i \in \hat{R}, \tag{17}$$

$$\sum_{j\in \hat{R}} x_{ji} = 1 \quad \forall i \in \hat{R}, \tag{18}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in \hat{A}, \tag{19}$$

where $\hat{R} = R \cup \{n+1, \ldots, n+nv\}$ is the set of nodes, $\hat{A} = \{(i,j) : i,j \in \hat{R}, i \neq j\}$ is the set of arcs and each arc (i,j) has associated a cost coefficient $\overline{p}_{ij}$. When i and j are both vehicles $\overline{p}_{ij} = \infty$. The values given to $\overline{p}_{ij}$, when $i \in V$ and $j \in N$ or vice versa is $\overline{p}_{ij} = t_{ij} + w_j$

It is easy to show that this assignment problem is a useful relaxation of the DARP. Indeed, the constraints involving the time windows, namely constraints (8) and (9), can be relaxed, since they have already been incorporated in the computation of $\overline{p}_{ij}$. Moreover, the minimization of the objective function takes care of the right part of constraints (9) (i.e., $p_i \leq l_i$). Constraints (4) and constraints (10) are irrelevant to this graph $\hat{G}$, hence removed. Constraints (2), (3), (5), (6) and (7) have to be taken into account. The last three constraints, ensuring the feasibility of the loads, are relaxed and the feasibility of the solution obtained is verified at a later stage. Moreover, thanks to the assumption $q_i = 1$, capacity usually is not one of the most stringent constraints. Constraints (2) and (3) are replaced by constraints (17) and (18).

The solution of problem (16)–(19) is a set of cycles. Some of these cycles, called main paths, contain several customers and the node depot and can be seen as chains of customers starting and ending with the node depot. The others, called subtours, contain only customers. The objective function leads the assignment towards a set of nv long chains of customers and a small number of subtours. Most of the "two customers" subtours are eliminated thanks to the fact that $\overline{p}_{ij} \neq \overline{p}_{ji}$. Longer subtours (three or more customers) occur very infrequently, since they are related to a set of customers close to each other temporally and spatially. The next section explains how the infeasible solution can be easily transformed in a feasible one, but before a second slightly different assignment problem is introduced below.

3.3.1 The "level of service" instead of the "departure time" as objective function

A tight relationship exists between p_i, the time windows associated to a customer i, respectively, $[e_{i^+}, l_{i^+}]$ and $[e_{i^-}, l_{i^-}]$, and S_i. Starting with the value e_{i^+}, the time windows are calculated as follows: $l_{i^-} = S^m * t_{i^+i^-} + e_{i^+}$; $e_{i^-} = t_{i^+i^-} + e_{i^+}$; and $l_{i^+} = l_{i^-} - t_{i^+i^-}$ (see also Amaldi et al. 2000). The value S^m represents the *level of service* guaranteed to the customer in the worst case: its trip cannot be longer than S^m times the minimum travel time. Therefore $S^m \geq S_i \geq 1$ and, recalling that $p_{i^-} = a_{i^-}$, the normalized value of S_i, indicated with $\tilde{S}$, is

$$\tilde{S}_i = \frac{p_{i^-} - e_{i^-}}{f_{i^-}}, \tag{20}$$

where $f_{i^-} = l_{i^-} - e_{i^-}$ is the width of the delivery time window and $1 \geq \tilde{S} \geq 0$.

The idea consists of calculating the $\tilde{S}$ for the two delivery nodes i^-, j^- in each sequence of four nodes starting with node i^+ (i.e., $\{i^+, j^+, i^-, j^-\}$, $\{i^+, i^-, j^+, j^-\}$, $\{i^+, j^+, j^-, i^-\}$). Then, the proposed procedure chooses the maximum between the two values, since it corresponds to the customer with the worst level of service in a two-customers sequence. This new value is defined as $\widetilde{S}^r_{ij} = \max(\tilde{S}_i, \tilde{S}_j)$ where r denotes which sequence, among the possible three ones, is under consideration and i, j represents the considered pair of customers. The average value is

$$\overline{S}_{ij} = \frac{\sum_{r=1}^{3} \widetilde{S}^r_{ij}\delta_r}{\sum_{r=1}^{3} \delta_r},$$

where $\delta_r = 1$ if $\widetilde{S}^r_{ij} < \infty$ and $\delta_r = 0$ otherwise. The value $\overline{S}_{ji}$ is calculated in the same way and $\overline{S}_{ij} \neq \overline{S}_{ji}$. When i and j are both vehicles $\overline{S}_{ij} = \infty$. The values given to $\overline{S}_{ij}$, when $i \in V$ and $j \in N$ or vice versa is $\overline{S}_{ij} = w_j$.

This value $\overline{S}_{ij}$ can be associated to arc (ij) instead of $\overline{p}_{ij}$. Thus, the assignment problem becomes

$$\min \sum_{(ij)\in\hat{A}} \overline{S}_{ij} x_{ij}$$

subject to constraints (17) and (18).

3.4 Make feasible an infeasible solution

The solution obtained with the assignment problem consists of a set of nv main paths and few subtours. The customers in the same subtour are close together both temporally and spatially, otherwise the algorithm will be unable to connect them together. To obtain a solution of the DARP, it is necessary to deal with the subtours and to make each one of the *nv* paths a feasible route (sequence of pickup and delivery nodes).

The nodes (clients) involved in the subtours are inserted in a simple way in the main paths: whenever a feasible insertion point exists the customer is inserted. Then, it is sufficient to transform the solution of the assignment problem for each vehicle into a feasible route. The solution of the assignment problem for each vehicle can be seen as a sequence of pick up nodes (e.g., $\{d, i^+, j^+, k^+, \ldots, d\}$). To obtain a route, it is necessary to insert the delivery nodes. A simple idea is to insert them in a straightforward and trivial way such as $\{d, i^+, i^-, j^+, j^-, k^+, k^-, \ldots, d\}$. The procedure starts inserting node i^-; if the arrival time in i^- is feasible (i.e., $p_{i^-} \leq l_{i^-}$), it goes on, otherwise it removes node i and considers node j. This goes on until the depot is reached and the same is done for the whole fleet of vehicles. The result is a set of v feasible trivial paths and a set of un-serviced customers. In the following, this procedure is called *trivial*.

The second procedure starts with the complete sequence of pick up and delivery nodes (e.g., $\{d, i^+, i^-, j^+, j^-, k^+, k^-, ..., d\}$). Then, it starts with the first delivery node (e.g., i^-) and tries to insert it in all the possible positions of the sequence starting after its pickup node (e.g., i^+). For each position the procedure checks only the local feasibility: if the time window of the delivery node under consideration is respected (i.e., $p_{i^-} \leq l_{i^-}$). The procedure stops to try to insert the delivery node when it arrives at the end of the route or it is clearly infeasible to go on due to the delivery node time window. Then, for each feasible position the procedure calculates the objective function for the whole route summing ∞ for each infeasibility in the sequence, and among all the feasible positions it chooses the cheapest. As in the procedure *trivial* customer i is removed from the sequence if there are no feasible insertion positions for the delivery node. Once the procedure has finish to evaluate the current delivery node (i^-) it starts again with the next (j^-) and it stops when all the delivery nodes have been evaluated. This procedure is called *smart*.

4 Computational results and conclusion

As far as the authors know, there are no benchmark instances for both the PDPTW and the DARP. For this reason the data set has been created using the complete Milan network (about 17,000 arcs and 7,000 nodes); these instances are proposed as benchmark problems for the future. The set of random instances have been created by extracting randomly a pair of nodes for each customer. Each customer requiring either the arrival time or the departure time. The time is generated randomly in a time interval of 240 min (4 h) and in the same way it is associated to the pickup node or to the delivery node. The machine used to solve the problem is a Pentium III with a 600 MHz processor and 256 MB of RAM. In each table, columns with the same label have the same meaning, which is the following. N is the number of customers, V the number of vehicles,

Table 1 Results obtained using $\overline{p}$ as objective function of the assignment problem

				Trivial		*Smart*	
N	V	I	Sub.(%)	S	Uns.(%)	S	Uns.(%)
10	2.1	(10)	0.0	0.040	11.0	0.131	3.0
20	3.0	(10)	0.0	0.088	18.5	0.209	8.5
30	3.4	(10)	4.7	0.106	22.3	0.285	13.0
40	4.0	(10)	4.0	0.107	19.8	0.275	10.3
50	4.5	(10)	8.0	0.127	25.6	0.286	18.6
60	5.0	(10)	6.5	0.165	26.0	0.321	15.3
70	5.4	(10)	6.6	0.176	28.6	0.316	19.1
80	6.0	(10)	10.1	0.164	32.3	0.310	22.5
90	6.0	(10)	8.2	0.171	35.2	0.336	23.8
100	7.0	(10)	6.6	0.199	32.7	0.330	20.9
110	7.2	(10)	7.5	0.201	35.8	0.341	25.5
120	7.3	(10)	10.6	0.220	40.8	0.359	29.3
130	7.9	(9)	8.5	0.213	39.1	0.327	29.0
140	8.2	(10)	9.5	0.223	40.0	0.364	28.9
150	8.7	(10)	9.7	0.222	41.0	0.359	28.5
160	9.0	(10)	9.9	0.229	40.8	0.356	28.9
170	9.2	(10)	10.8	0.238	42.6	0.367	30.6
180	9.4	(10)	11.4	0.250	44.3	0.365	33.4

I the number of instances generated and solved, *Sub.* the number of customers involved in the subtours, *Uns.* the number of unserved customers (the number of customers removed to make the solution feasible), S the average level of service. All the tables provide average values out of the number of instances. The computation time is omitted in all the tables, since it is less than a $\frac{5}{100}$ of second (≤ 0.05) on average for each instance solved with the total set of four algorithms.

Tables 1 and 2 report the computational results with the set of data generated: the number of customers goes from 10 to 180. The results have been obtained without inserting the subtours in the main paths (i.e., without step 5 in the framework), since the purpose of this first comparison is to understand which procedure, among the four presented, is the best one without any additional help. Table 1 presents the results of the algorithm with $\overline{p}$ as objective function. Table 2 presents the results of the algorithm with $\overline{S}$ as objective function. The results obtained with the procedure (*trivial*) are reported in columns 5 and 6, while columns 7 and 8 reports those obtained with the (*smart*) procedure. The quality of the solution is measured in terms of customers served, since the fleet of vehicles is fixed. For the same number of customers the average level of service must be optimum. The *smart* algorithm is always better than the *trivial* one, as expected. The solution obtained with $\overline{p}$ as objective function of the assignment problem seems often better than the one obtained with $\overline{S}$ in terms of number of satisfied customers. The solution obtained with $\overline{S}$ as objective function are competitive for the small problem (10 and 20 customers), while for larger instances $\overline{p}$ gives always better solutions. When the number of custom-

Table 2 Results obtained using $\overline{S}$ as objective function of the assignment problem

				Trivial		*Smart*	
N	V	I	Sub.(%)	S	Uns.(%)	S	Uns.(%)
10	2.1	(10)	0.0	0.043	11.0	0.118	3.0
20	3.0	(10)	0.7	0.048	25.3	0.144	15.0
30	3.4	(10)	1.0	0.082	26.3	0.224	13.7
40	4.0	(10)	2.9	0.105	23.8	0.209	16.5
50	4.5	(10)	4.5	0.093	31.8	0.211	22.4
60	5.0	(10)	4.5	0.121	35.5	0.230	26.0
70	5.4	(10)	4.4	0.122	37.0	0.265	25.7
80	6.0	(10)	7.8	0.123	40.1	0.268	28.8
90	6.0	(10)	10.3	0.151	42.1	0.280	32.4
100	7.0	(10)	11.0	0.122	41.5	0.233	31.0
110	7.2	(10)	8.9	0.150	42.2	0.274	33.5
120	7.3	(10)	13.5	0.164	45.8	0.292	37.3
130	7.9	(9)	10.6	0.139	46.3	0.282	36.6
140	8.2	(10)	14.9	0.152	46.7	0.290	38.4
150	8.7	(10)	15.4	0.168	47.6	0.303	38.1
160	9.0	(10)	16.5	0.167	47.9	0.281	40.1
170	9.2	(10)	19.1	0.146	50.8	0.292	40.6
180	9.4	(10)	20.9	0.180	52.1	0.309	41.6

Table 3 A comparison between the proposed algorithm and a straightforward insertion heuristic

		Smart		Insertion	
N	V	Uns.(%)	S	Uns.(%)	S
10	2	3	0.18	5	0.27
30	3	10	0.33	15	0.42
60	5	12	0.35	18	0.45
90	6	20	0.42	27	0.49
120	7	25	0.43	33	0.50
150	9	27	0.44	35	0.52
180	9	30	0.47	39	0.53

ers inserted is equal, it is clear that $\overline{S}$ gives solutions with a better S and that the smart procedure increases the number of customers served, decreasing the quality of S. The total number of removed customer is less than 20% for the first set of instances (from 10 to 100 customers). Whereas for larger instances (from 110 to 180 customers) the number of removed customers is always less or equal to 30% except that for the biggest instances.

Table 3 compares the results obtained with the best of the proposed algorithms (i.e., the (*smart*) procedure with $\overline{p}$ as objective function) with those obtained by a simple straightforward insertion procedure. For this comparison the complete described algorithm has been used (step 5 included). The comparison has been done on a subset of the proposed instances: 10, 30, 60, 90, 120, 150, 180. The number of vehicles is fixed. The proposed algorithm finds always a better solution: in terms of served customers it is able to satisfy from 3% up to 8% plus on average and with a lower value of S, despite the larger number of clients on board. The computation time is less than 0.005 s, therefore omitted. These results shown that the proposed heuristic is fast and effective, since it obtains better results than a simple insertion heuristic in a similar computation time. The difference increases when the algorithms have been tested on a real-time (on-line) instances with time-dependent network. In this case the proposed algorithm gives better solutions and at the same time it is 30% faster (on average).

References

Amaldi E, Colorni A, Fiorenzo Catalano S (2000) Feasibility study of a dial-a-ride system for a suburban area of milan. Ric Oper, 30(94–95):5–28

Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9: 375–382

Cordeau J-F (2003) A branch and cut algorithm for a dial-a-ride problem. Oper Res (in press)

Cordeau J-F, Laporte G (2003a) The dial-a-ride problem (darp): variants, modeling issues and algorithms. 4OR 1:89–101

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

Cordone R, Wolfler Calvo R (1996) Note about time window constraints in routing problems. Internal report 96-005. Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano

Desrosiers J, Dumas Y, Soumis F (1986) A dynamic programming solution of the large-scale single-vehicle Dial-a-Ride problem with time windows. Am J Math Manag Sci 6

Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Network routing. Handbooks in operations research and management science, vol 8, pp 35–139. North-Holland

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

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

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

Kontoravdis G, Bard JF (1995) A GRASP for the vehicle routing problem with time windows. ORSA J Comput 7:10–23

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

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulations and traveling salesman problems. J Assoc Comput Mach 7:326–329

Psaraftis HN (1980) A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transp Sci 14(2):130–154

Psaraftis HN (1983) An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transp Sci 17(3):351–357

Savelsbergh MWP (1985) Local search in routing problems with time windows. Ann Oper Res 4:285–305

Savelsbergh MWP, Sol M (1995) The general pickup and delivery problem. Transp Sci 29(1):17–29

Sexton T, Choi Y (1986) Pick-up and delivery of partial loads sith time windows. Am J Math Manag Sci 6:369–398

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

Wolfler Calvo R (2000) A new heuristic for the traveling salesman problem with time windows. Transp Sci 34(1):113–124