Available online at www.sciencedirect.com
ScienceDirect
Transportation Research Part B 41 (2007) 736–755
TRANSPORTATION RESEARCH PART B
www.elsevier.com/locate/trb

A rejected-reinsertion heuristic for the static Dial-A-Ride Problem

Ying Luo, Paul Schonfeld *

Department of Civil and Environmental Engineering, University of Maryland, College Park, 1173 Glenn L. Martin Hall, College Park, MD 20742, United States

Received 19 August 2005; accepted 1 February 2007

Abstract

We present here a new heuristic for the static multi-vehicle Dial-A-Ride Problem, which we call a rejected-reinsertion heuristic. Its main objective is to minimize the number of vehicles used to satisfy all the demand, subject to service quality constraints. Passenger deviation from desired time is minimized in the scheduling stage after the insertion position is determined. This method improves the basic parallel insertion heuristic in two aspects. First, a rejected-reinsertion operation is performed each time it is infeasible to insert a new request into the vehicle routes. Each assigned request close to the new request in time frame and geographic location is tentatively removed from its current vehicle and the new request is inserted into the best position in that vehicle route, followed by the reinsertion of the removed request elsewhere in the system. Of all available rejected-reinsertions, the least-cost one is then implemented. Second, an improvement procedure including trip reinsertion and trip exchange operations is implemented periodically. Two sets of problems are tested in a computational study. These show that the proposed heuristic achieves vehicle reductions of up to 17% over the parallel insertion heuristic and is very efficient computationally.

Keywords: Dial-A-Ride; Heuristic algorithm; Paratransit service; Scheduling; Public transportation

1. Introduction

Dial-A-Ride (DAR) paratransit is one of the public transit services which can provide shared-ride door-to-door service with flexible routes and schedules. In a Dial-A-Ride Problem (DARP), passengers specify transportation requests with their origins, destinations, and desired pickup or delivery times. A set of routes and schedules must be determined to best accommodate the demand under a set of constraints. The two most common service quality constraints relate to maximum time deviation from desired time, i.e., the difference between the actual pickup/delivery time and the desired pickup/delivery time, and maximum ride time, i.e. the actual passenger in-vehicle time. DAR generally provides a higher quality of service (e.g. negligible access

* Corresponding author. Tel.: +1 301 405 1954; fax: +1 301 405 2585.
E-mail addresses: ying.luo@gmail.com (Y. Luo), pschon@umd.edu (P. Schonfeld).

0191-2615/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trb.2007.02.003

time, wait at home and no transfers) but increases operating cost due to a lower vehicle productivity (e.g. passenger trips per vehicle hour) than conventional bus services.

DAR is intended to provide intermediate service between buses and taxis in terms of both operating cost and level of service. As shared-ride taxi, the cost per user is reduced at some sacrifice in level of service. The productivity and level of service depend very much on efficiency of routing and scheduling. In the existing systems or algorithms, two types of service requests are considered: advance requests and real-time requests. The advance requests usually refer to those received at least one day before the service is provided, so that routes and schedules can be planned before the start of the service. Real-time requests are those asking for same-day service either as soon as possible or at specified times. If all the requests are advance requests (and assuming all other factors, such as traffic conditions, are predictable), then the determination of the routes and schedules is a static DARP; otherwise, the problem becomes a dynamic problem, in which the routes and schedules must be determined in real-time. Solutions of static DARPs often constitute bases for dynamic DARPs as the latter are solved as a sequence of static problems.

In this paper we describe an insertion-based rejected-reinsertion heuristic for the multi-vehicle static DARP with service quality constraints, with special emphasis on minimizing the number of vehicles that satisfies all the demand, thus maximizing vehicle productivity. This study analyzes a static problem, in which all the demands are known at the time when the vehicle routes are planned. Most current DAR services for the elderly and disabled operate in the static mode. The proposed heuristics have been adapted to solve the dynamic DARP (Luo, 2006).

The DARP is a generalization of the Pickup and Delivery Problem (PDP) and the Vehicle Routing Problem (VRP), which are known to be NP-hard. The DARP is a PDP in which the loads to be transported represent people. Usually in the DARP maximum ride time constraints are considered and time windows are narrower than those in the PDP. The DARP is different from and somewhat more difficult than most other routing problems due to the precedence and maximum time constraints, the tight time window constraints and also because operator cost and user inconvenience must be weighted against each other when designing a solution instead of considering only operator cost. For a DARP overview, see Cordeau and Laporte (2003). Other related overviews include Bodin et al. (1983) for general routing and scheduling of vehicles and crews, Desrosiers et al. (1995) and Solomon (1987) for vehicle routing and scheduling problems with time window constraints, and Desaulniers et al. (2002), Mitrovic-Minic (2001) and Savelsbergh and Sol (1995) for the general PDP. The following review focuses on the scientific literature specific to the static DARP.

Due to the NP-hard nature of the DARP, its solution methods are almost exclusively heuristic except for very small single-vehicle problems. Psaraftis (1980, 1983a) develops an exact dynamic programming algorithm for the single-vehicle static problem without and with time windows. Less than 10 customers are considered in his example. Desrosiers et al. (1986) solve the single-vehicle problem by formulating it as an integer problem and solving it exactly by dynamic programming. Instances with up to 40 users have been solved. Other heuristic approaches for the single-vehicle problem include the work by Psaraftis (1983b) and Sexton and Bodin (1985a,b).

The heuristics for the multi-vehicle DARP can be classified primarily into four general categories: insertion-based, cluster-first route-second, metaheuristics and improvement procedure. An insertion-based algorithm inserts one passenger request into the vehicle routes at a time, at a position that is feasible and results a minimum increase of a pre-specified objective function. The general insertion-based method includes work by Jaw et al. (1986), Kikuchi and Rhee (1989), Madsen et al. (1995), Potvin and Rousseau (1992), Toth and Vigo (1997) and Diana and Dessouky (2004)'s regret insertion method. The basic idea of the insertion method was applied in other studies with special objectives (Dessouky et al., 2003; Fu, 2002, 2003).

Cluster-first route-second is a commonly used technique in various VRPs and has been attempted in the DARP. It is usually embedded with the mathematical programming technique for solving the clustering and/or routing phases. Sexton and Bodin (1985a,b) apply Benders' decomposition procedure to a mixed binary nonlinear formulation of the static single vehicle problem, which separates the routing and scheduling components allowing each to be attacked individually. Bodin and Sexton (1986) further develop a cluster first, route and schedule second and swap the third heuristic for the multi-vehicle problem. Desrosiers et al. (1988) solve the multiple-vehicle DARP by mini-clustering first, routing second. Ioachim et al. (1995) improve the mini-clustering phase by using a mathematical optimization technique to form the mini-clusters and solving

the problem by column generation. Borndorfer et al. (1997) use a set partitioning approach for the solution of the problem. Baugh et al. (1998) approach the problem by using simulated annealing for clustering and a modified space-time nearest neighbor heuristic for developing the routes for the clusters.

In the third category, Cordeau and Laporte (2003) use a tabu search heuristic for the static DARP, which is extended by Attanasio et al. (2004) for the dynamic version by using the parallel computing technique. Toth and Vigo (1997) also develop a tabu thresholding procedure, which can improve the solution obtained by their insertion heuristic. Hart (1996) develops a simulated annealing based solution heuristic for the Dial-A-Ride Problem. The heuristic is computationally expensive (e.g. a 30 or 40 customer problem will require thousands of seconds).

The improvement category includes the work of Van Der Bruggen et al. (1993) who develop a local search method for the single-vehicle pickup and delivery problem with time windows based on a variable-depth search, and work by Toth and Vigo (1996) who describe local search refining procedures which can be used to improve the solutions for large problems obtained by a parallel insertion heuristic.

Insertion heuristics have proved to be popular methods for solving a variety of vehicle routing and scheduling problems because they are fast, can produce fair solutions, are easy to implement, and can easily be extended to handle complicating constraints (Campbell and Savelsbergh, 2004). A comparative study by Solomon (1987) indicates the insertion method is an effective heuristic for the vehicle routing problem with time windows, especially for heavily time-constrained problems. The cluster-first route-second approach is difficult to apply to the DARP since the cluster phase needs special considerations due to the DARP pairing and time window constraints. Metaheuristics such as tabu search are computationally very expensive and their performance is directly related to running time and calibration of the algorithm parameters. Also, for a heavily constrained problem such as DARP, it is very difficult to maintain the feasibility in each local neighbor movement, or converge to a final feasible solution if infeasibility is allowed during local movements.

In this paper, we describe a new insertion-based rejected-reinsertion heuristic, which keeps the fast computation advantage of an insertion method, while improving the algorithm performance mainly in terms of the number of vehicles required to serve all the demand. Technological advances such as Advanced Vehicle Location (AVL), Global Position Systems (GPS), Geographical Information Systems (GIS) and similar systems are making real-time dispatching more feasible. Our insertion-based method has potential for application in a dynamic context due to its computational efficiency and improved performance.

The remainder of the paper is organized as follows. Section 2 describes the basic operating scenario of the DAR service. Section 3 presents the proposed rejected-reinsertion heuristic, in which the rejected-reinsertion operator, improvement procedure, variable fleet size, feasibility check of inserting and removing a request, and scheduling are discussed. In Section 4, two sets of problems are tested and the results are summarized. Section 5 contains some concluding remarks.

2. Operating scenario of the service

In this paper, the operating scenario is similar to the one described by Jaw et al. (1986). More specifically,

1. Each passenger i specifies *either* a desired pick-up time DPT_i at his/her origin *or* a desired delivery time DDT_i at his/her destination.
2. Deviation constraint from desired time: A passenger with a desired pick-up time will be picked up during time period $[DPT_i, DPT_i + TW_i]$ and a passenger with a desired delivery time will be delivered during time period $[DDT_i - TW_i, DDT_i]$. TW_i is the pre-specified maximum deviation from desired time and it is usually the same for all the passengers.
3. Ride time constraint: A passenger's actual ride (in-vehicle) time will not exceed a given maximum ride time MRT_i, which is usually a function of the passenger's direct ride time DRT_i.
4. A vehicle is not allowed to wait idly while carrying passengers.
5. Vehicle capacity should not be violated. In the DAR context, due to the low vehicle productivity, the vehicle capacity is usually not a relevant constraint.

The level of service is guaranteed by the constraint on deviation from desired pickup or delivery time and by the maximum ride time constraint, which limit the worst case bounds for the service quality. In other words, a passenger's actual time deviation would not exceed the maximum time deviation, and a passenger's actual ride time would not exceed the maximum ride time set by the level of service. The average service quality will be better than those bounds allow. The fourth constraint assures that the passengers do not sit in an idle vehicle during their trips just waiting for other passengers (except to board or exit), which would deteriorate the DAR service quality.

The deviation constraint from desired time and maximum ride time constraint are usually transferred into time windows for pickup and delivery to facilitate the feasibility check for the insertion (Jaw et al., 1986). We define EPT_i and LPT_i as the earliest and latest pickup times for request i, and EDT_i and LDT_i as the earliest and latest delivery times for request i.

For customers specifying desired pickup time (DPT):

$$EPT_i = DPT_i \tag{1a}$$
$$LPT_i = EPT_i + TW_i \tag{1b}$$
$$EDT_i = EPT_i + DRT_i \tag{1c}$$
$$LDT_i = LPT_i + MRT_i \tag{1d}$$

For customers specifying desired delivery time (DDT):

$$LDT_i = DDT_i \tag{2a}$$
$$EDT_i = LDT_i - TW_i \tag{2b}$$
$$LPT_i = LDT_i - DRT_i \tag{2c}$$
$$EPT_i = EDT_i - MRT_i \tag{2d}$$

3. Proposed rejected-reinsertion heuristic

Before proceeding further, the basic parallel insertion algorithm (Jaw et al., 1986) is summarized since it is the basis of the proposed algorithm.

Consider N passenger requests for service and M available DAR vehicles. The parallel insertion algorithm (Jaw et al., 1986) first sorts the passengers in sequence (i.e. based on their earliest pickup times). Then each customer is processed in the list in sequence, and assigned to a vehicle until the list of customers is exhausted.

For each customer i ($i = 1, 2, \ldots, N$),

Step 1: For each vehicle j ($j = 1, 2, \ldots, M$)
- (a) Find all the feasible insertion sequences in which customer i can be inserted into the work-schedule of vehicle j. If it is infeasible to assign customer i to vehicle j, examine the next vehicle $j + 1$, and restart Step 1; Otherwise.
- (b) Find the insertion of customer i into the work-schedule of vehicle j that results in minimum additional cost. Call this additional cost C_j.

Step 2: If it is infeasible to insert i into any vehicle j, then declare a "rejected customer"; otherwise, assign to the vehicle j^* for which $C_{j^*} \leqslant C_j$ for all j ($j = 1, 2, \ldots, M$).

Denote n as the maximum number of trips in a route. Then each parallel trip insertion requires $O(n^2N)$ time. In fact both stops of each trip can be inserted into $O(n^2)$ different positions in a route and thus can be inserted into $O(n^2M)$ different positions in all routes. The feasibility check for a given sequence of stops can be computed in $O(n)$ time. Therefore, each parallel trip insertion requires $O(n^2N)$ time and the parallel insertion heuristic (for all requests) requires $O(n^2N^2)$ time.

Algorithm variations exist, depending mostly on the sorting scheme, insertion criteria and the determination of the vehicle schedules once an insertion sequence is determined. We sort the passengers by their earliest pickup times. Insertion criteria and vehicle scheduling will be discussed in later sections.

3.1. Rejected-reinsertion operator

The main disadvantage of the insertion method is that it works in a myopic way in that each request is inserted into its current best position without having an overview of all the requests. The regret insertion heuristic (Diana and Dessouky, 2004) alleviates the problem by calculating for each unassigned request its regret, which is a measure of the potential cost that could be paid if the given request were not immediately inserted, and inserting the request with the largest regret. Local improvement procedures such as swapping the customers into different routes or reinserting the customer could also improve the routing and scheduling in terms of an explicit objective function (i.e. total vehicle travel distance).

The basic idea of the rejected-reinsertion operation can be illustrated in Fig. 1 using a simplified scenario. Consider the scenario in Fig. 1a with two vehicle routes and two new requests 1 and 2 to be scheduled. '+' and '−' represent the origin and destination of a request, respectively. Assume request 1 has the earlier pickup time so that it will be scheduled first. Also assume that it is feasible to insert request 1 into either route 1 or route 2. Using the basic insertion method, request 1 is inserted into route 2 which produces a smaller insertion cost, as shown in Fig. 1b. When turning to schedule request 2, we might find that it is infeasible to insert request 2 into route 2 because the schedule of route 2 during the time windows of request 2 is filled. It might also not be inserted into route 1 because it is too far from route 1 to make that insertion feasible. Under this condition, either request 2 is rejected or more vehicles are needed.

In the case in Fig. 1, whenever an infeasible insertion occurs (e.g., insertion of request 2), we attempt to vacate some slot for the infeasible request by removing another request that is similar in terms of time frame and geographic location from its current route and reinserting it into some other route. If the new request can be inserted into the available vacancy and the removed request can be reinserted somewhere else, then the insertion algorithm proceeds to schedule the next request. If either of the requests cannot be inserted, the above search is repeated with another previously assigned request. A deeper search, incorporated in all the heuristics tested below, considers all previously assigned requests, instead of stopping after finding the first feasible one. The "least cost" set of moves is selected for implementation. In Fig. 1c, by the rejected-reinsertion operation, request 1 is removed from route 2 and reinserted into route 1 and request 2 is inserted into route 2. In this way, some of the myopic behavior of the insertion method is alleviated. The concept of rejected-reinsertion is simple and straightforward but is very effective in reducing the number of vehicles used, as will be shown in the computational study. The detailed procedure for the rejected-reinsertion operation is as follows:

Assume that requests up to $k-1$ have been scheduled. For new request k, if it is infeasible to insert the new request,

1. For each request $i = 1,\ldots,k-1$.
2. If request i and request k satisfy time proximity criterion 1 defined as

   $EPT_i \leqslant LDT_k$ and $EPT_k \leqslant LDT_i$, goto Step 3; else goto Step 1.
3. Remove request i from its planned route R_i, and calculate the associated removal cost as C^i_{remove}. (It is actually a saving and the value should be negative.)
4. Insert request k into route R_i. If it is feasible, calculate the associated insertion cost as C^k_{insert}, and go to Step 5; else go to Step 1.
5. Insert request i considering all the available vehicles. If it is feasible, calculate the associated insertion cost as C^i_{insert}, and the total cost $C_{\text{total}} = C^i_{\text{remove}} + C^k_{\text{insert}} + C^i_{\text{insert}}$.
6. Go to Step 1.
7. Make the move with the minimum total cost C^*_{total}.

Note that it is still possible that a request may be infeasible to schedule. It will then be rejected or served by an additional vehicle.

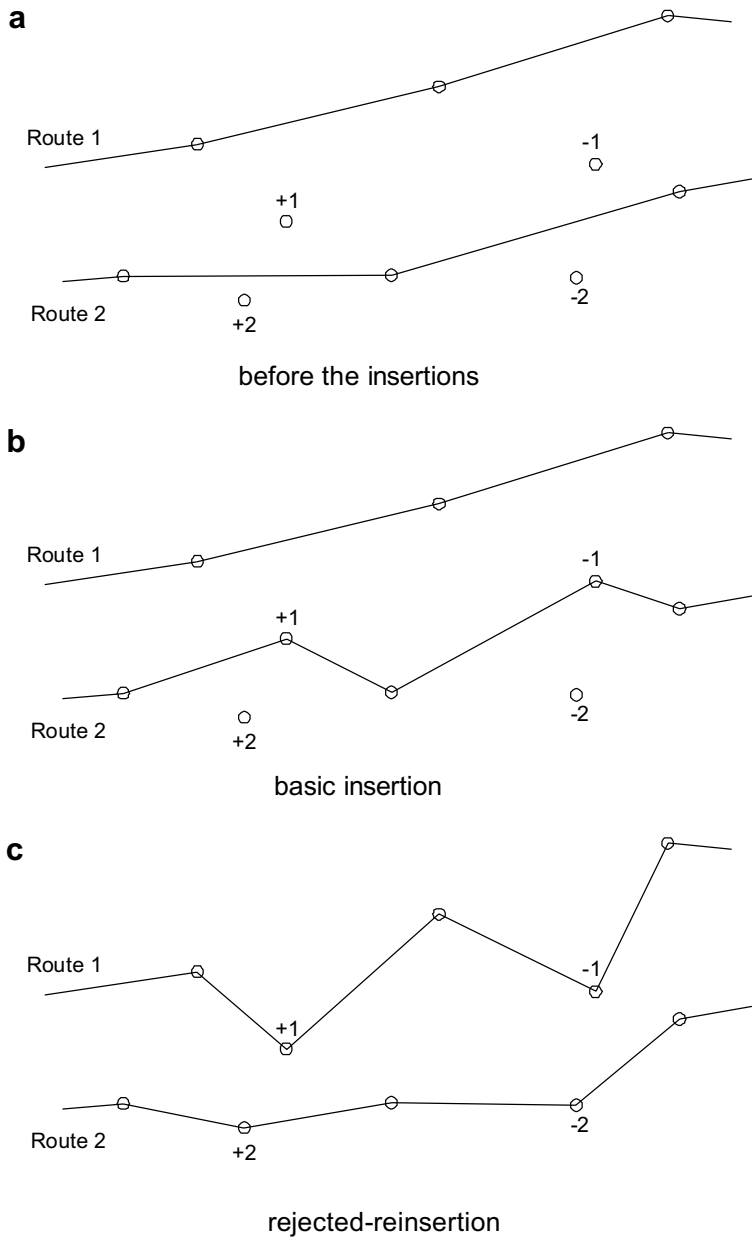

Fig. 1. Illustration of the rejected-reinsertion method: (a) before the insertions; (b) basic insertion; (c) rejected-reinsertion.

Each rejected-reinsertion operation requires $O(n^2N^2)$ time since all N passengers must be evaluated for removal and reinsertion and each removal and reinsertion needs $O(n^2N)$ time. Therefore, the proposed rejected-reinsertion procedure requires $O(n^2N^3)$ time in the worst case in which each request needs a rejected-reinsertion operation. However, by setting an appropriate initial fleet size, the computation time can be reduced since only a small fraction of the requests need the rejected-reinsertion operation.

3.2. Improvement procedure

One option of the heuristic is to add a local improvement procedure periodically after a certain number of insertions or at interval Δ (e.g. the improvement procedure can be applied after insertion of the ith request if

$EPT_i < k\Delta \leqslant EPT_{i+1}$, $k = 1, 2, \ldots$). Two inter-route reassignment operators (Toth and Vigo, 1996) are considered in the local improvement procedure: (1) *Trip reinsertion* operator: remove trip i from its current route and reinsert it into all the vehicle routes (the final route could be the same as the current one); (2) *Trip exchange* operator: remove trip i from its route r and remove trip j from its route s; insert the two stops of trip i in the best positions of route s and insert the two stops of trip j in the best positions of route r.

The implementation of one iteration of the trip reinsertion is as follows: examine all assigned requests in sequence; when a trip reinsertion yields a total cost below zero, apply it and examine the next assigned request. The time complexity of one trip reinsertion iteration is $O(n^2N^2)$.

Due to the high computational cost, the trip exchange operation is performed only on the restricted neighborhoods. For one iteration of the trip exchange procedure, we examine assigned requests i from 1 to $N-1$. Only those assigned requests j ($j = i+1, \ldots, N$) are considered for exchange that satisfy time proximity criterion 2 defined as:

$$\text{pickup time window overlap } EPT_i \leqslant LPT_j \quad \text{and} \quad EPT_j \leqslant LPT_i,$$
$$\text{delivery time window overlap } EDT_i \leqslant LDT_j \quad \text{and} \quad EDT_j \leqslant LDT_i,$$

Whenever the trip exchange results a total cost less than zero, the trip exchange operation is implemented. Theoretically, the time complexity of an iteration of the trip exchange is $O(n^2N^3)$ since there are $O(N^2)$ exchange combinations with $O(n^2N)$ time for each reinsertion. However, the above defined time proximity criterion 2 will limit the number of exchange combinations due to the highly constrained nature of the problem. The implementation of the improvement procedure consists of iterating the trip exchange procedure until no further improvement is possible, followed by the iteration of the trip reinsertion procedure until no further improvement is possible. The whole procedure is repeated until no change occurs or some prescribed number of iterations is reached. Based on our computational experiments, the whole procedure will usually stop after 2–4 iterations.

In the extreme, the improvement interval Δ could be the whole service period, which means the improvement procedure will be applied after all requests have been inserted. However, for a large-scale DARP, the improvement procedure over all the requests is extremely time consuming. Furthermore, an improvement procedure applied periodically is more likely to assign the requests compactly than one applied after all requests have been inserted. Therefore, the periodical improvement is applied in this study and the sensitivity of the improvement interval is analyzed in Section 4.1.5.

3.3. Variable vs fixed fleet size

In order to satisfy all the demand, either a sufficient fleet size should be provided initially if fleet size is fixed throughout the planning process, or fleet size should be increased during the insertion process to serve otherwise infeasible demand. In the former case, the minimum number of vehicles required to serve all the demand is usually obtained by tentatively using different numbers of vehicles and running the algorithm repeatedly in order to find the minimum number which satisfies all the demand. The algorithms with variable and fixed fleet size will be compared in the computational study. The initial fleet size for the variable fleet size will also be tested.

3.4. Complete heuristic procedures

The complete rejected-reinsertion heuristic procedure can be described as follows:

1. Set the initial fleet size F_0.
2. Sort the passengers in the order of their earliest pickup times.
3. Insert the passengers in sequence. For each passenger, if insertion into the current fleet is infeasible, perform the rejected-reinsertion operation specified in Section 3.1. If it is still infeasible to insert the passenger into the current fleet, add one new vehicle into the fleet and insert the passenger into it. The new vehicle can serve all subsequent requests.

The complete rejected-reinsertion heuristic with improvement procedure can be described as follows:

1. Set the initial fleet size F_0.
2. Sort the passengers in the order of their earliest pickup times.
3. Group the passengers into time bins with Δ duration in each bin according to their earliest pickup times (i.e. The earliest pickup times of the first group fall between $[0, \Delta]$; the earliest pickup times of the second group fall between $[\Delta, 2\Delta]$; ...; the earliest pickup times of the kth group fall between $[(k-1)\Delta, k\Delta]$; ... and so on).
4. Insert the passengers in sequence. For each passenger, if insertion into the current fleet is infeasible, perform the rejected-reinsertion operation specified in Section 3.1. If it is still infeasible to insert the passenger into the current fleet, add one new vehicle into the fleet and insert the passenger into it. The new vehicle can serve all subsequent requests.
5. Perform the improvement procedure after insertion of each group of passengers.

The kth improvement procedure is applied to those passengers in the kth group as well as those passengers whose desired service times are close to the newly inserted group of passengers (i.e. those passengers whose $LPT_i > (k-1)\Delta$).

The setting of the initial number of vehicles F_0 in Step 1 is not essential. Sensitivity analysis in the computational study will show that the resulting minimum number of vehicles required to serve all the demand is quite insensitive to F_0. Basically, the value of F_0 should be less than the required number of vehicles to serve all the demand.

3.5. Feasibility check for inserting a request

For each insertion of the origin and destination stops of a request, all the constraints including those on vehicle capacity, time windows and maximum ride time of all passengers should be satisfied. If n is the maximum number of the trips in a vehicle route and M is the number of operating vehicles, the number of possible insertions is of $O(n^2M)$. Jaw et al. (1986) proposed four statistics to expedite the time window feasibility check (which is the most difficult and time-consuming). A "schedule block" (SB) concept was first proposed by Jaw et al. (1986) in facilitating the feasibility check of each attempted insertion. This concept applies to the version of DARP in which no vehicle can be idle while there are passengers onboard. It is defined as a continuous period of active vehicle time between two successive periods of vehicle slack (idling) time, starting and ending with empty vehicle. For each stop α within schedule block k they define four statistics BUP_α, $BDOWN_\alpha$, AUP_α and $ADOWN_\alpha$ to facilitate the feasibility check of inserting the origin and destination of a request into the same schedule block. BUP_α ($BDOWN_\alpha$) represents the maximum amount of time by which stop α and all its preceding stops in the same schedule block can be advanced (delayed) without violating the time window constraints. AUP_α ($ADOWN_\alpha$) similarly represents the maximum amount of time by which stop α and all its following stops can be advanced (delayed).

The statistics can only be used when inserting the origin and destination of a request into the *same* schedule block. The maximum shifts are bounded by the available slack times at the ends of the schedule block. However, the insertion of the origin and destination of a request should not be unnecessarily constrained to the same schedule block, especially for such a highly constrained problem. The statistics can be easily generalized for the case considering the whole route instead of one schedule block, as follows:

$$BUP_i = \begin{cases} \min(BUP_{i-1} + Idle_k, AT_i - ET_i) & \text{if stop } i \text{ is the first stop of one schdule block } k \\ \min(BUP_{i-1}, AT_i - ET_i) & \text{otherwise} \end{cases}$$

$$BDOWN_i = \begin{cases} LT_i - AT_i & \text{if stop } i \text{ is the first stop of one schedule block } k \\ \min(BDOWN_{i-1}, LT_i - AT_i) & \text{otherwise} \end{cases}$$

$$AUP_i = \begin{cases} AT_i - ET_i & \text{if stop } i \text{ is the last stop of one schedule block } k \\ \min(AUP_{i+1}, AT_i - ET_i) & \text{otherwise} \end{cases}$$

$$ADOWN_i = \begin{cases} \min(ADOWN_{i+1} + Idle_{k+1}, LT_i - AT_i) & \text{if stop } i \text{ is the last stop of one schedule block } k \\ \min(ADOWN_{i+1}, LT_i - AT_i) & \text{otherwise} \end{cases}$$

In the above definition, BUP_i ($BDOWN_i$) represents the maximum amount of time by which stop i and all its preceding stops in the same *vehicle route* can be advanced (delayed) without violating the time window constraints. AUP_i ($ADOWN_i$) represents the maximum amount of time by which stop i and all its following stops can be advanced (delayed). AT_i, ET_i and LT_i are the actual, earliest and latest times (either pickup or delivery) for stop i, respectively. $Idle_k$ is the idling (slack) time before schedule block k. If pickup stop $+i$ of a new request is inserted between stop p and $p+1$ and delivery stop $-i$ is inserted between stops p and $p+1$, then the necessary time window feasibility conditions include:

$$T_p + BDOWN_p + T_{p,+i} \geqslant EPT_i \quad \text{if stop } p \text{ is not the last stop of one schedule block} \tag{3}$$

$$T_p - BUP_p + T_{p,+i} \leqslant LPT_i \tag{4}$$

$$T_{q+1} + ADOWN_{q+1} - T_{-i,q+1} \geqslant EDT_i \tag{5}$$

$$T_{q+1} - AUP_{q+1} - T_{-i,q+1} \leqslant LDT_i \quad \text{if stop } q \text{ is not the last stop of one schedule block} \tag{6}$$

$$T^i_{\text{detour}} \leqslant BUP_p + ADOWN_{q+1} + Idle_{p+1,q+1} \tag{7}$$

In Eqs. (3)–(6), T_i denotes the scheduled time for stop i and $T_{i,j}$ denotes the direct ride time from stop i to stop j. In Eq. (7), $Idle_{p+1,q+1}$ is the total idling time between stop p and $q+1$. T^i_{detour} is the additional travel time due to inserting both stops $+i$ and $-i$.

$$T^i_{\text{detour}} = T_{p,+i} + T_{+i,-i} + T_{-i,p+1} - T_{p,p+1} \quad \text{if} \quad p = q \tag{8a}$$

$$T^i_{\text{detour}} = T_{p,+i} + T_{+i,p+1} + T_{q,-i} + T_{-i,q+1} - T_{p,p+1} - T_{q,q+1} \quad \text{if} \quad p \neq q \tag{8b}$$

Note that the idling time between stops $p+1$ and $q+1$ should be eliminated if idling is not permitted while passengers are onboard. If insertion of both the origin and destination of a request are feasible in terms of time window constraints, the maximum ride time constraints of assigned passengers (and the capacity constraint if necessary) should also be checked by scanning through the list of customers and comparing the attempted ride times with the maximum ride times.

3.6. Feasibility check for removing a request

The rejected-reinsertion, trip reinsertion and trip exchange operations all involve the removing of a request from its assigned route. Special caution should be exercised when removing a request from its current route because it might violate a window for some other passengers on the route. This only applies to the operating scenario considered in which a vehicle is not allowed to idle while carrying passengers. Fig. 2 shows one complete schedule block from which one request is to be removed. For illustration purposes, only removal of one stop (stop b) will be discussed and only the time windows of some stops are shown in Fig. 2. Assuming stop a would not be the last stop of a possible new schedule block after stop b is removed from the route, then stop c should be visited directly from stop a without idling. If there is more time between stop a and c than the direct ride time, the stops preceding stop a could be pushed forward and/or the stops following stop c could be

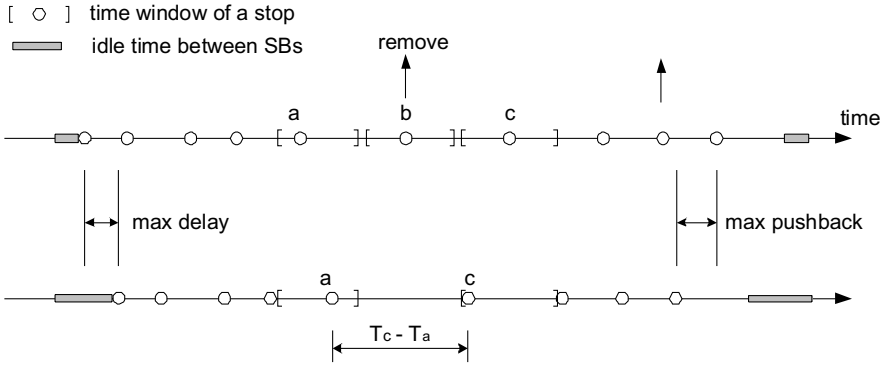

Fig. 2. Removing a request from one schedule block.

pushed backward to reduce the time gap between a and c. In Fig. 2, the 'max delay' is the maximum amount of time that all stops preceding stop b could be delayed without violating the time window constraints, and the 'max pushback' is the maximum amount of time that all stops following stop b could be pushed backward. If the direct travel time from stop a to c (plus the service time at stop a, if that is considered) is less than the time interval between stops a and c (i.e., $T_c - T_a$ in Fig. 2), then idling time before stop c is necessary or the time window constraints of some stops within the current schedule block will be violated if no idling time between a and c is provided. Thus, the removal of a request may cause a time window violation.

When the time window violation occurs in the removal process, the route may become feasible again if at least one stop is inserted into the same schedule block (i.e. after stop a) in the reinsertion step (i.e. the insertion of the previously rejected passenger into the removed route in the rejected-reinsertion operation).

3.7. Insertion criterion

In the insertion heuristic, the insertion decision is made based on the additional increase of the objective function. The insertion with the least incremental cost will be chosen. In the context of service operations in the public sector, there are always tradeoffs between minimizing the operating cost and the passenger inconvenience cost. A general form of the objective function might include active vehicle travel time (moving time)/distance, excess ride time (the difference between the actual ride time and direct ride time) of all current passengers, time deviation (the difference between the actual pickup/delivery time and desired time) of all current passengers and vehicle idling time.

Although selecting the weights of components is up to the system operating managers and the proposed heuristic does not depend on the objective form chosen, the ultimate objective here is to minimize the number of vehicles required in order to maximize the vehicle productivity, which is usually very low for DAR systems due to their high quality of service (i.e. door-to-door service) and dispersed demand. Also, because the passenger inconvenience (i.e. waiting time, excess ride time) is already formulated as hard constraints, it seems unnecessary to include it in the objective function at the cost of more vehicles used. However, the number of vehicles is an input to the algorithm and cannot be expressed in the objective function explicitly. A common alternative way is to minimize the vehicle travel time/distance. Some studies (i.e. Jaw et al., 1986) implicitly suggest including other components, such as vehicle idling time, in the objective function, as that reserves some flexibility for future demand. Thus, the components in the objective function work more like heuristic parameters.

3.8. Vehicle scheduling

Scheduling refers to the determination of the actual pickup and delivery times of the new insertion and the corresponding modification of the actual pickup and delivery times of the affected passengers assigned once the insertion sequence is determined. The scheduling will affect the passenger time deviation, but will not affect the passenger ride time and vehicle travel time/distance. The schedules can be formed as soon as possible (Diana and Dessouky, 2004), or can be optimized based on the incremental cost (Jaw et al., 1986). For a congested system, the two methods may lead to similar results. Our experimental tests show that the above two scheduling methods achieve very similar results. In this study, schedules are sought that minimize the time deviation of the passengers. A more detailed discussion of schedule optimization based on general cost functions can be found in Jaw (1984).

4. Computational study

Although static DARPs have been studied by many researchers, there are very few benchmark problems available for comparison. One reason might be that there is far less research on DARPs than on general VRPs. Another reason is that different operational scenarios (i.e. whether or not vehicles are allowed to be idle while carrying passengers) or objectives are considered for different studies, which further reduces the available test problems in each category.

Below, we test our heuristics with our own randomly generated problems and with test problems from Diana and Dessouky (2004). The latter problems are the latest found in the literature that consider operational

scenarios very similar to ours. The randomly generated problems have smaller service areas and average direct travel distances compared with the second set of problems. Although both problem categories consider the time-dependent demand, in the randomly generated problems, the demand is relatively stable, which might justify the usage of the same fleet size throughout the service period. To deal with the randomness of the demand, five replications are generated for each problem, and the statistics reported are the average over five replications. The computer program is coded using visual C++ and is run on a personal laptop with a 1.6 GHz Pentium M and 768M of RAM.

4.1. Randomly generated problems

An 8 mile × 8 mile service area with the depot located in the center of the area is studied. The Euclidean distance metric is used with a circuity factor of 1.3 (by which each direct distance is multiplied). Vehicle speed is assumed to be constant at 15 mph. The locations of origins and destinations of all the demand are uniformly and independently distributed in the area. The time intervals between consecutive earliest pickup times follow a negative exponential distribution. We simulate 9 h of service with the hourly demand as 120, 120, 160, 200, 200, 160, 160, 120, 120 requests per hour. The departure times from and return times to the depot are not restricted to the 9-h period. Vehicle capacity is assumed to be a large number. The maximum number of passengers onboard simultaneously will be recorded, which indicates the minimum vehicle size should be provided. Four service quality scenarios as constrained by time window and maximum ride time are considered. A linear maximum ride time equation is used as follows:

$$MRT = a + b \cdot DRT \tag{9}$$

Table 1 shows the parameter settings for the four scenarios 'LL', 'L', 'M' and 'H'. The service quality improves from 'LL' to 'H'.

We include active vehicle travel time (when the vehicle is moving) and passenger excess ride time in the objective function. The component of the passenger excess ride time works somewhat like a heuristic parameter. Based on some experimental tests, we found that the minimum number of vehicles used is not very sensitive to the weight assigned to the passenger excess ride time component as long as that weight is below 0.5 for the two sets of problems. For the passenger excess ride time, a weight of 0.2–0.3 yields slight better solutions than a weight of zero. The values of the weights used in this study are 0.7 for the active vehicle travel time and 0.3 for the passenger excess ride time.

For each scenario considered, six algorithm variations are implemented and compared. Algorithm 1 is the basic parallel insertion heuristic similar to that of Jaw et al. (1986) except that insertions across multiple schedule blocks are allowed and insertion schedules are determined to minimize the time deviation from the desired time. The fleet size is fixed throughout the planning process. Algorithm 2 is similar to Algorithm 1. The difference is that one vehicle is added to the fleet whenever it is infeasible to insert a new request into the current fleet. Algorithm 3 differs from Algorithm 1 in that rejected-reinsertion is implemented for those rejected requests. Algorithm 4 combines features of Algorithms 2 and 3, in which the rejected-reinsertion is implemented for rejected requests and fleet size is added after a request is rejected by the rejected-reinsertion operation. In Algorithms 2w and 4w, a periodical improvement procedure at 30-min time interval is applied to Algorithms 2 and 4. The starting fleet sizes for Algorithms 2, 2w, 4 and 4w are 30, 40, 50 and 65 for scenarios LL, L, M and H, respectively. For Algorithms 1 and 3, the number of vehicles required is obtained by running the program repeatedly using different fleet sizes and finding the smallest one that satisfies all the demand.

Table 1
Constraint settings for four service quality scenarios

Scenario	Time window (min)	Constant term a in Eq. (9) (min)	Slope b in Eq. (9)
LL	30	5	2.5
L	20	5	2.0
M	10	5	1.5
H	5	5	1.3

4.1.1. Test of rejected-insertion heuristic without and with periodical improvement

Table 2 reports detailed statistics for Algorithms 1, 4 and 4w as follows. 'Vehicle miles' is the total vehicle travel distance in miles. 'Vehicle prod.' is the vehicle productivity defined as the number of served passengers divided by the total vehicle service time (including idling time). The sixth column reports the total passenger miles. The average passenger time deviation from the desired times and average passenger ride ratio are reported in the next two columns. 'Max passengers onboard' indicates the vehicle capacity required since a large vehicle capacity is assumed. Finally, the last column indicates the average computation time in seconds.

Based on Table 2, Algorithm 4 outperforms Algorithm 1 in terms of number of vehicles (up to −9.7%) and vehicle productivity (up to +5.8%) at a cost of slightly increased passenger time deviation and ride time ratio. The vehicle productivity increases as the number of vehicles decreases. The average passenger time deviation is slightly less than half of the maximum deviation from desired time. As constraints become more restrictive, Algorithm 4 becomes increasingly superior in solution quality to Algorithm 1. Algorithm 4 is still very efficient computationally although its computation time is approximately doubled in the H scenario and quintupled in the LL scenario compared to Algorithm 1.

Algorithm 4w further improves on the results of Algorithm 1 in terms of number of vehicles (−9.7% to −16.6%), vehicle miles (−2.6% to −12.3%), vehicle productivity (+6.7% to +15.7%), passenger miles and ride time ratio. The improvement is more prominent for the LL and L scenarios than for the M and H scenarios. This occurs because the DARP is a heavily constrained problem, and as the problem gets more restricted, the feasible region for improvement becomes more limited. This conclusion is based on scenarios in which vehicles are already heavily loaded. It is expected that if vehicles are less loaded or time windows are wider, the improvement will be greater but would require much more computation time. As the problem gets less constrained (from H to LL), the computation time increases nonlinearly.

4.1.2. Test of rejected-reinsertion operator

Table 3 shows the minimum number of vehicles required for all demand that results from six algorithm variations. In Table 3, Algorithms 1 and 3, 2 and 4, 2w and 4w are comparison pairs. The rejected-reinsertion operator is implemented in the second algorithm of each pair. It is found that the rejected-reinsertion operator used in Algorithms 3, 4 and 4w is very effective in reducing the vehicle fleet for all four scenarios.

4.1.3. Test of fixed vs variable fleet size

Still in Table 3, comparing Algorithm 1 with 2, and 3 with 4, Algorithm 2 (variable fleet size) slightly underperforms Algorithm 1 (fixed fleet size) in the LL scenario, but slightly outperforms it in the H scenario. Algorithm 4 (variable fleet size) performs similarly with Algorithm 3 (fixed fleet size), except that in the H scenario, Algorithm 4 succeeds with slightly fewer vehicles. While the differences are small, a common trend is that as the problem gets more restricted, algorithms with variable fleet sizes become more preferable. One advantage of the algorithm using variable fleet size over the one using fixed fleet size is that there is no need to run the algorithm repeatedly each time trying a different fleet size and finding the smallest one that satisfies all demand. The advantage becomes more relevant for an algorithm which needs more computation time (i.e. an algorithm with a periodical improvement procedure).

4.1.4. Variability of the solutions

Table 4 reports the variability of the solutions in terms of number of vehicles required for the 30 randomly generated replications. This variability of the solutions arises, in part, from the randomness of the demand generation process and highly constrained nature of the problem. On the other hand, it shows how robust the heuristics are in terms of the solution consistency. The minimum, maximum, average and standard deviation of the fleet size required for the four service scenarios are shown in the last four columns of Table 4. The results show that the standard deviation decreases as the problems get less restrictive. The standard deviations through Algorithms 4 and 4a are consistently smaller than that through Algorithm 1 and standard deviation through Algorithm 4a is generally smaller than that through Algorithm 4. This means that the proposed rejected-reinsertion heuristics are more robust than the basic parallel insertion heuristic in handling the random demand in a highly constrained environment. This also suggests that the fleet size requirement might be predicted from a given level of demand and level of service (in terms of maximum time deviation and maxi-

Table 2
Comparison of three algorithm variations

Scenario	Algorithm	# Of vehicles	Vehicle miles	Vehicle prod.	Passenger miles	Average deviation (min)	Ride time ratio	Max passengers onboard	Comp. time (s)
LL	1	40.2	4 813	3.95	11 328	14.21	1.475	10.2	16
	4	38.4	4 791	4.09	11 859	14.06	1.546	9.4	83
	4 vs 1	−4.5%	−0.5%	+3.5%	+4.7%	−1.1%	+4.8%		
	4w	34.0	4 223	4.57	10 862	14.69	1.420	9.6	1 686
	4w vs 1	−15.4%	−12.3%	+15.7%	−4.1%	+3.4%	−3.7%		
L	1	49.4	5 323	3.40	10 414	8.89	1.362	8.0	13
	4	45.6	5 339	3.55	10 670	9.00	1.396	8.2	40
	4 vs 1	−7.7%	+0.3%	+4.4%	+2.5%	+1.2%	+2.5%		
	4w	41.2	4 769	3.86	10 074	9.54	1.323	7.8	497
	4w vs 1	−16.6%	−10.4%	+13.5%	−3.3%	+7.3%	−2.9%		
M	1	62.2	6 242	2.73	9 308	4.30	1.231	5.6	11
	4	58.0	6 298	2.85	9 394	4.26	1.243	5.8	23
	4 vs 1	−6.8%	+0.9%	+4.4%	+0.9%	−0.9%	+1.0%		
	4w	55.4	5 885	3.00	9 252	4.51	1.224	6.2	126
	4w vs 1	−10.9%	−5.7%	+9.9%	−0.6%	+4.9%	−0.6%		
H	1	78.2	7 101	2.23	8 589	1.85	1.146	4.2	11
	4	70.6	7 135	2.36	8 627	1.92	1.151	4.2	19
	4 vs 1	−9.7%	+0.5%	+5.8%	+0.4%	+3.8%	+0.4%		
	4w	70.6	6 916	2.38	8 601	1.95	1.147	5.0	62
	4w vs 1	−9.7%	−2.6%	+6.7%	+0.1%	+5.4%	+0.1%		

Table 3
Comparison of algorithms with and without rejected-reinsertion operator

Algorithm	# Of vehicles required for each scenario			
	LL	L	M	H
1	40.2	49.4	62.2	78.2
2	43.8	50.2	63.4	76.4
2w	37.0	44.0	58.8	75.8
3 (3 vs 1)	37.6 (−6.5%)	45.4 (−8.1%)	58.4 (−6.1%)	72.8 (−6.9%)
4 (4 vs 2)	38.4 (−12.3%)	45.6 (−9.2%)	58.0 (−8.5%)	70.6 (−7.6%)
4w (4w vs 2w)	34.0 (−8.1%)	41.2 (−6.4%)	55.4 (−5.8%)	70.6 (−6.9%)

Table 4
Variability of number of vehicles over 30 replications

Scenario	Algorithm	Minimum	Maximum	Average	Standard deviation
LL	1	36	45	40.6	2.19
	4	35	41	38.7	1.80
	4w	31	37	34.0	1.43
L	1	43	59	48.8	3.11
	4	41	48	45.0	1.88
	4w	40	46	41.1	1.69
M	1	57	70	62.3	3.08
	4	53	65	58.1	2.31
	4w	50	61	55.4	2.44
H	1	69	84	76.7	3.63
	4	66	81	70.7	3.30
	4w	65	74	69.2	2.91

mum ride time ratio). Luo (2006) developed performance models (i.e. for fleet size requirement, average passenger waiting time and average passenger ride time ratio) for DAR systems using a simulation metamodeling approach.

4.1.5. Sensitivity analysis

The rejected-reinsertion heuristics do not involve complex algorithm parameters which must be fine-tuned for specific problems. The heuristic parameters include the initial fleet size and the improvement interval if improvement procedure is implemented. The sensitivity of those parameters is tested in this section.

Table 5 shows the fleet size required to serve all demand through Algorithms 4 and 4w, using different initial fleet sizes. The results are found to be quite insensitive to the initial fleet size. In general, using an initial fleet size close to the required fleet size achieves slightly better results (and also requires fewer rejected-reinsertion operations) than using a smaller initial fleet size. The required fleet size can be easily estimated by running the algorithm once using any reasonable initial fleet size.

Table 6 shows the number of vehicles required when we vary the improvement interval (15, 30, 60 and 90 min) at which the improvement procedure is implemented (Algorithm 4w). The results show that solutions through Algorithm 4w are not quite sensitive to the improvement interval. Using smaller improvement interval achieves slight better results. However, using larger improvement intervals generally requires more computation time as the number of eligible requests for reinsertion and exchange increases in each improvement procedure. (As noted in Section 3.4, the kth improvement procedure is applied to passengers in the kth group as well as those passengers whose desired service times are close to the newly inserted group of passengers (i.e. passengers whose $LPT_i > (k-1)\Delta$). Therefore, for those passengers the improvement procedure might be applied more than once. This is why the computation time of instances whose improvement

Table 5
Effects of initial fleet size on final fleet size with Algorithms 4 and 4w

Scenario	Algorithm	Initial fleet size					
		10	20	30	40	50	65
LL	4	38.4	38.8	38.4			
	4w	35.0	34.6	34.0			
L	4		45.8	45.6	45.6		
	4w		42.4	42.4	41.2		
M	4			58.4	59.4	58.0	
	4w			57.0	56.8	55.4	
H	4				71.6	71.8	70.6
	4w				69.8	70.2	70.6

Table 6
Sensitivity to the improvement interval

Improvement interval (min)	# Of vehicles (computation time in seconds)			
	LL	L	M	H
15	34.0 (2 013)	41.2 (553)	55.4 (126)	70.4 (70)
30	34.0 (1 686)	41.2 (497)	55.4 (126)	70.6 (62)
60	34.2 (1 854)	42.6 (480)	57.0 (128)	70.4 (69)
90	35.4 (2 355)	42.4 (613)	57.2 (132)	71.6 (67)

interval is 15 min is larger than that when improvement interval is 30 min). An improvement interval of 30 min is generally appropriate and is used in this study.

Therefore, the proposed rejected-reinsertion heuristics are simple and robust in the sense that they do not involve complex algorithm parameters and do not require much fine-tuning for their parameters.

4.2. Diana and Dessouky's test problems

The main purposes of testing the second set of problems are to check that our algorithms are correctly implemented using computer language and to compare their performances with other available sources.

4.2.1. Input data characteristics

The test problems of Diana and Dessouky (2004) include one 500-request problem and one 1000-request problem, each with five replications. The data are randomly generated, but based on data provided by a realistic DAR system run by Access Services, Inc. For example, the distribution from which the pickup times of the samples were drawn was based on the empirical distribution derived from Los Angeles County. Interested readers may refer to Diana and Dessouky (2004) and Dessouky and Adam (1998) for more information on the data generation. In this paper the 1000-request problem is tested and used to compare algorithms.

The basic operational scenario is briefly summarized as follows:

Total service area: 150 mile × 150 mile.

Vehicle speed: 15 mph (the number has been corrected by the author during our correspondence).

Probability of serving a wheelchair passenger: 0.2.

Service time distribution: uniform (1, 3) minutes for wheelchair passengers;

30 s for others.

Simulation period: 0:00–23:59.

Fig. 3 shows the demand distributions over time for one replication of the 500-request problem and one replication of the1000-request problem. Fig. 4 shows the direct travel time distribution for one of the replications of the 1000-request problem.

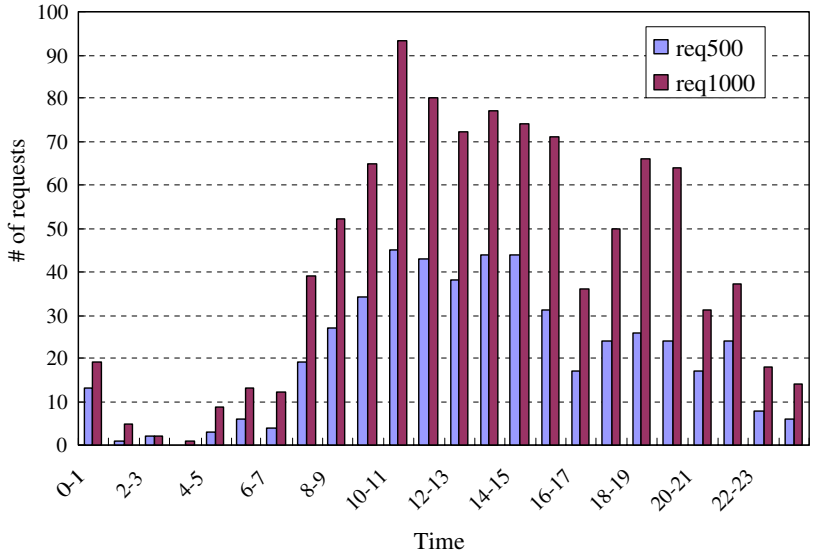

Fig. 3. Demand distribution over time.

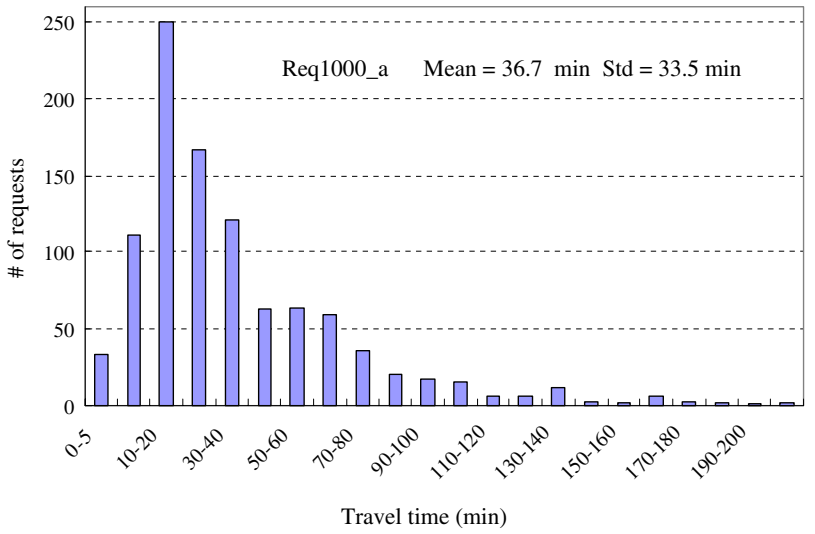

Fig. 4. Direct travel time distribution.

Table 7
Constraint settings for three service quality scenarios

Scenario	Time window (min)	Constant term a in Eq. (9) (min)	Slope b in Eq. (9)
L	30	20	2.0
M	15	10	1.5
H	5	5	1.2

For the 1000-request problem, three scenarios 'L', 'M' and 'H' are tested, whose constraint settings are defined in Table 7. (In Diana and Dessouky (2004), one base scenario was tested too, in which the service quality is between M and H.) Note that although the service qualities defined here are very similar to those defined in Table 1 for the randomly generated problems, the problems here are more constrained than the randomly

generated ones. This occurs because the average direct travel time is longer and the area covered is larger in the problems defined by Diana and Dessouky (2004) than in our randomly generated problems.

4.2.2. Computational results

In Tables 8–10, Algorithms D1 and D5 correspond to Algorithms 1 and 5 in Diana and Dessouky (2004), which represent the basic parallel insertion algorithm (into same schedule block) and their proposed regret insertion algorithm (across multiple schedule blocks and schedule as soon as possible). For comparability, the same objective function is used; thus, the weights for vehicle travel distance, passenger excess ride time and vehicle idle times within the schedule are 0.45, 0.50 and 0.05. The definition of the time windows by Diana and Dessouky includes the stop service time, while ours does not. Their maximum ride time constraint is interpreted as the sum of the actual ride time and of the service times at the pickup and delivery stops must not exceed the maximum ride time, while in ours the service times are not counted in the maximum allowable ride time. We have adjusted those small discrepancies in the problem definitions to make results comparable.

Tables 8–10 shows the computational results for the 1000-request problem under the L, M and H scenarios. '# of vehicles', 'Vehicle miles', 'Ride time ratio' and 'Comp. time' are as previously defined. 'Idle hours' reports the total length of all the vehicle idling times. Note that values of vehicle miles for D1 and D5 in Table 8–10 have been adjusted due to a rounding problem based on correspondence with one of the authors.

Table 8
Computational results for L scenario of 1000-request problem

Algorithm	# Of vehicles	Vehicle miles	Idle hours	Ride time ratio	Comp. time (s)
D1	63.2	15675	288	1.395	n/a
D5	58.4	14820	301	1.476	n/a
1	60.8	13917	138	1.193	8
(1 vs D1)	(−3.8%)	(−11.2%)	(−52.1%)	(−14.5%)	
4	52.2	13788	97	1.271	16
4w	51.6	13402	104	1.214	74
(4w vs D5)	(−11.6%)	(−9.6%)	(−65.4%)	(−17.8%)	

Table 9
Computational results for M scenario of 1000-request problem

Algorithm	# Of vehicles	Vehicle miles	Idle hours	Ride time ratio	Comp. time (s)
D1	77.2	17 655	350	1.173	n/a
D5	70.0	16 530	374	1.204	n/a
1	72.0	15 811	220	1.102	9
(1 vs D1)	(−6.7%)	(−10.4%)	(−37.1%)	(−6.1%)	
4	66.4	15 771	200	1.105	11
4w	65.6	15 462	200	1.101	32
(4w vs D5)	(−6.3%)	(−6.5%)	(−46.5%)	(−8.6%)	

Table 10
Computational results for H scenario of 1000-request problem

Algorithm	# Of vehicles	Vehicle miles	Idle hours	Ride time ratio	Comp. time (s)
D1	92.8	20 160	464	1.034	n/a
D5	87.2	19 110	485	1.042	n/a
1	91.6	18 386	368	1.022	8
(1 vs D1)	(−1.3%)	(−8.8%)	(−20.7%)	(−1.2%)	
4	86.8	18 443	346	1.023	15
4w	86.6	18 376	348	1.022	24
(4w vs D5)	(−0.7%)	(−3.8%)	(−28.2%)	(−1.9%)	

In comparing Algorithms 1 and D1, both of which are basic parallel insertion heuristics but with variable and fixed fleet size, we find that Algorithm 1 uses similar numbers of vehicles for the H scenario but slightly fewer vehicles for the L and M scenarios than Algorithm D1. However, Algorithm 1 outperforms D1 in terms of total vehicle miles, idle hours and ride time ratio. The large reduction of idle times by Algorithm 1 may be due to the use of a smaller initial fleet size. Since the demand level is low during the early service period and thus few vehicles are needed, using larger fleet size will increase idle time.

Algorithm 4 outperforms Algorithm 1, as in the randomly generated test cases described earlier. Comparing Algorithm 4w (rejected-reinsertion with periodical improvement) with D5 (regret insertion), 4w outperforms D5 with up to 11.6% fewer vehicles used in the L scenario. Its advantage decreases as the service quality increases (i.e. the problem gets more restricted). In their computational test, Diana and Dessouky (2004) found the regret insertion algorithm to perform better with medium to small time window constraints. Note that the advantage of Algorithm 4w over Algorithm 4 is relatively limited in this set of test problems compared to the set of randomly generated problems, since this set of problems is more restrictive in that 1000 requests are distributed in a very large area (i.e. 150 mile $\times$ 150 mile), thus complicating the scheduling. For those more restrictive problems, Algorithm 4 gets similar results as Algorithm 4w, but with faster computation.

Note that the ride time ratio in Table 10 is very low, even though the constant and slope terms for the maximum ride time (Eq. (9)) for this scenario are 5 min and 1.5, respectively. The obtained average ride time ratio is far below half of the maximum ride time ratio. The vehicle occupancy is around 0.51 for Algorithms 1, 4 and 4w. (It is not reported for Algorithms D1 and D5.) As the constraint on deviation from desired time and the maximum ride time constraint get more restrictive, more vehicles are required and vehicle productivity and vehicle occupancy decrease. The indicated tradeoffs between the vehicle resources and service quality (i.e. average time deviation and excess ride time) should be very useful to DAR planners.

The proposed heuristic is very efficient computationally. Without the periodical improvement procedure, Algorithm 4 solves a 1000-request problem within 16 s. The computation time for algorithms with the periodical improvement increases as the problem gets less restricted. The low quality case of the 1000-request problem takes about 74 s. (The computational times reported by Diana and Dessouky (2004) are 26 min for the 500-request problems and 195 min for the 1000-request problems on a Pentium III computer.) The computation times of the proposed heuristic are clearly fast enough for practical applications.

5. Conclusions

In this paper, we propose a rejected-reinsertion heuristic for the multi-vehicle DARP with service quality constraints. The main innovation of the heuristic is a rejected-reinsertion operator. Whenever the insertion of a new request is infeasible, this operator persists in inserting it by trying to move previously assigned requests elsewhere. The least-cost set of moves is determined and implemented. The insertion process is tested with fixed and variable fleet sizes. A periodical improvement procedure involving trip reinsertion and trip exchange is also implemented to further improve the solution.

Through the computational study, the proposed heuristic is shown to be effective, especially in reducing the number of required vehicles and thus increasing vehicle productivity. The rejected-reinsertion heuristic without periodical improvement can achieve moderately better results than parallel insertion heuristics for all cases studied. The rejected-reinsertion heuristic with periodical improvement outperforms the parallel insertion heuristic by using up to 17% fewer vehicles. Among the problems considered here, the periodical improvement procedure is more effective for the less constrained ones. The heuristic still maintains the advantages of an insertion-based method whose computational performance is quite good and which can be extended to a dynamic problem. Using a variable rather than fixed fleet size does not change the results much, but it eliminates the trial-and-error process for obtaining the minimum required fleet size.

Based on its performance on the DARP studied in this paper, the proposed rejected-reinsertion operator seems promising for other vehicle routing problems with time windows, especially for heavily time-constrained problems (e.g., PDP or taxi scheduling). This operator alleviates the myopic behavior of an insertion method in an efficient way. The quality and computational efficiency of the heuristic also make it attractive for application in dynamic problems. The proposed rejected-reinsertion heuristic is still applicable to dynamic problems where customers may request service with different advance times (the time difference between the

request time and the desired service time). This kind of system is believed to be more cost-effective than one serving immediate requests in that better routing and scheduling could be achieved by exploiting the available advance information for trip reinsertion and/or exchange.

Further research should extend the heuristic to dynamic versions of the problem, including cargo deliveries, and test its performance in dynamic applications.

Acknowledgements

The authors wish to thank Dr. Marco Diana and Dr. Maged M. Dessouky for providing the datasets and helping clarify some data issues. They also wish to thank two anonymous reviewers for their constructive comments.

References

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

Baugh, J.W., Kakivaya, G.K.R., Stone, J.R., 1998. Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. Engineering Optimization 30, 91–123.

Bodin, L.D., Golden, B., Assad, A., Ball, M., 1983. Routing and scheduling of vehicles and crews: the state of the art. Computers and Operations Research 10 (2), 63–211.

Bodin, L.D., Sexton, T., 1986. The multi-vehicle subscriber dial-a-ride problem. TIMS Studies in the Management Sciences 22, 73–86.

Borndorfer, R., Grotschel, M., Klostermeier, F., Kuttner, C., 1997. Telebus Berlin: vehicle scheduling in a dial-a-ride system. Computer-Aided Transit Scheduling. In: Lecture Notes in Economics and Mathematical Systems, vol. 471. Springer-Verlag, Berlin, pp. 391–422.

Campbell, A.M., Savelsbergh, M., 2004. Efficient insertion heuristics for vehicle routing and scheduling problems. Transportation Science 38 (3), 369–378.

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

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M.M., Soumis, F., 2002. VRP with pickup and delivery. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, pp. 225–242.

Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., 1995. Time constrained routing and scheduling. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), . In: Network Routing, Handbooks in Operations Research and Management Science, vol. 8. Elsevier Science, Amsterdam, pp. 35–139.

Desrosiers, J., Dumas, Y., Soumis, F., 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences 6, 301–325.

Desrosiers, J., Dumas, Y., Soumis, F., 1988. The multiple vehicle dial-a-ride problem. Computer-Aided Transit Scheduling. In: Lecture Notes in Economics and Mathematical System, vol. 308. Springer, Berlin, pp. 15–27.

Dessouky, M., Adam, S., 1998. Real-time scheduling of demand responsive transit service – final report. University of Southern California, Department of Industrial and Systems Engineering, Los Angeles.

Dessouky, M., Rahimi, M., Weidner, M., 2003. Jointly optimizing cost, service, and environmental performance in demand-responsive transit scheduling. Transportation Research Part D 8, 433–465.

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

Fu, L., 2002. Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. Transportation Research Part B 36, 485–506.

Fu, L., 2003. Analytical model for paratransit capacity and quality-of-service analysis. Transportation Research Record 1841, 81–89.

Hart, S.M., 1996. The modeling and solution of a class of dial-a-ride problems using simulated annealing. Control and Cybernetics 25 (1), 131–157.

Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M.M., Villeneuve, D., 1995. A request clustering algorithm for door-to-door handicapped transportation. Transportation Science 29 (1), 63–78.

Jaw, J.J., 1984. Heuristic algorithms for multi-vehicle, advance-request dial-a-ride problems. Ph.D. Dissertation, Department of Aeronautics and Astronautics, MIT, Cambridge, MA.

Jaw, J.J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.M., 1986. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation Research Part B 20 (3), 243–257.

Kikuchi, S., Rhee, J.H., 1989. Scheduling method for demand-responsive transportation system. Journal of Transportation Engineering 115 (6), 630–645.

Luo, Y., 2006. Heuristics and performance metamodels for the dynamic dial-a-ride problem. Ph.D. Dissertation, Department of Civil and Environmental Engineering, University of Maryland, College Park.

Madsen, O.B.G., Ravn, H.F., Rygaard, J.M., 1995. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Annals of Operations Research 60, 193–208.

Mitrovic-Minic, S., 2001. The dynamic pickup and delivery problem with time windows. Ph.D. Dissertation, School of Computing Science, Simon Fraser University, Burnaby, Canada.

Potvin, J.Y., Rousseau, J.M., 1992. Constraint-directed search for the advanced request dial-a-ride problem with service quality constraint. In: Balci, O., Sharda, R., Zenios, S.A. (Eds.), Computer Science and Operations Research: New Developments in Their Interfaces. Pergamon Press, Oxford, England, pp. 457–474.

Psaraftis, H.N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science 14 (2), 130–154.

Psaraftis, H.N., 1983a. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transportation Science 17 (3), 351–357.

Psaraftis, H.N., 1983b. Analysis of an O(N^2) heuristic for the single vehicle many-to-many Euclidean dial-a-ride problem. Transportation Research Part B 17, 133–145.

Savelsbergh, M.W.P., Sol, M., 1995. The general pickup and delivery problem. Transportation Science 29 (1), 17–29.

Sexton, T., Bodin, L.D., 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. Transportation Science 19, 378–410.

Sexton, T., Bodin, L.D., 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: II. routing. Transportation Science 19, 411–435.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35 (2), 254–265.

Toth, P., Vigo, D., 1996. Fast local search algorithms for the handicapped persons transportation problem. In: Osman, I.H., Kelly, J.P. (Eds.), Meta-Heuristics: Theory and Applications. Kluwer, Boston, pp. 677–690.

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

Van Der Bruggen, L.J.J., Lenstra, J.K., Schuur, P.C., 1993. Variable-depth search for the single-vehicle pickup and delivery problem with time windows. Transportation Science 27 (3), 298–311.