Model and Algorithm Considering Time-Varying Travel Times to Solve Static Multidepot Dial-a-Ride Problem

Taehyeong Kim and Ali Haghani

This paper studies a static dial-a-ride problem (DARP) with time-varying travel times, soft time windows, and multiple depots. A static DARP model is formulated as a mixed-integer programming problem. To validate the model, several random small network problems are solved by using the commercial optimization package CPLEX. Three heuristic algorithms based on sequential insertion, parallel insertion, and clustering first–routing second are proposed to solve this problem within a reasonable time for implementation in a real-world situation. The results of the three heuristic methods are compared with the results obtained from exact solution by CPLEX to validate and evaluate the three heuristic algorithms. Computational results show that the three heuristic algorithms are superior to the exact algorithm in relation to the calculation time as the problem size (in number of demands) increases. Of the three heuristic algorithms, the one based on sequential insertion is more efficient than the other heuristic algorithms based on parallel insertion and clustering first–routing second.

Dial-a-ride service (also called demand responsive service or paratransit) is the most widely available transit service in the United States, with over 5,000 agencies providing it. Dial-a-ride service is composed of passenger cars, vans, or small buses operating in response to calls from passengers or their agents to a transit operator. The operator dispatches a vehicle to pick up the passengers and transport them to their destinations. Most agencies limit this service to disabled persons, their attendants and companions, or seniors.

The dial-a-ride problem (DARP) is nondeterministic polynomial time hard, belongs to the generic class of vehicle routing and scheduling problems, and has been extensively studied over the past 40 years (*1*). In the DARP, customers specify the desired pickup location, the drop-off location, the number of passengers, and time windows at the pickup location, the drop-off location, or both. The basic aim is to plan a set of minimum-cost vehicle routes capable of accommodating as many requests as possible under a set of constraints. The model objectives and constraints have variations. Most DARPs use a general cost function formed to combine several objectives, such as minimizing total travel time, the number of vehicles used, customers' excess ride time, customers' waiting time, and customers' service time deviations, together as an objective function. Furthermore, most DARPs include several constraints that must be satisfied by each route: round trip, coupling, precedence, vehicle capacity, time window, route duration, maximum ride time, and maximum waiting time.

Department of Civil and Environmental Engineering, University of Maryland, College Park, MD 20742. Corresponding author: A. Haghani, haghani@umd.edu.

Transportation Research Record: Journal of the Transportation Research Board, No. 2218, Transportation Research Board of the National Academies, Washington, D.C., 2011, pp. 68–77.
DOI: 10.3141/2218-08

Several surveys on models and algorithms developed for DARPs are described by Savelsbergh and Solomon, Mitrovic-Minic, Desaulniers et al., and Cordeau and Laporte (*2–7*). Generally, all demands are known in advance on the basis of reservations on previous days or subscriptions for regular service in static DARPs. Usually, this problem needs to be solved before the operations start at the beginning of every day.

For static multivehicle DARPs, Jaw et al. proposed advanced DARP with time windows and developed one of the first insertion heuristics by means of sequential insertion for the DARP (*8*). It was applied to randomly generated instances containing 250 requests and a real world instance with 2,617 requests. Ioachim et al. proposed a minicluster-first, route-second approach by using column generation to solve a multivehicle handicapped transportation system with time windows (*9*). Toth and Vigo examined the problem of determining an optimal schedule for a fleet of vehicles used to transport handicapped persons with time windows in an urban area and developed a parallel insertion procedure for their problem (*10*). To improve this heuristic procedure, they developed a tabu thresholding postoptimization procedure. Baugh et al. presented a heuristic algorithm that used a clustering-first–routing-second strategy for solving multiple-vehicle DARPs (*1*). They solved the problem by using simulated annealing for clustering and a modified space–time nearest-neighbor heuristic for developing the routes for the clusters. In addition, a tabu list was included to improve the performance of the simulated annealing algorithm. Before Fu (*11, 12*), all researchers assumed that travel times in an urban traffic environment were fixed and constant. He proposed improving paratransit scheduling by considering dynamic and stochastic variations in travel time. He found that both dynamic and stochastic variations in travel times had important effects on the quality of the schedules and that an appropriate consideration of these variations in the scheduling process could substantially improve the reliability and productivity of the schedules. Cordeau and Laporte proposed a tabu search heuristic for the static multivehicle DARP (*13*). Diana and Dessouky presented a new regret insertion heuristic for solving large-scale DARPs with time windows (*14*). This algorithm is a parallel-insertion heuristic with regret metric aimed at improving the myopic behavior that is often the drawback of insertion algorithms. A rejected-reinsertion heuristic for the same problem was proposed by Luo and Schonfeld (*15*). Xiang et al. used a local search strategy based on an insertion algorithm, a diversification strategy, and an intensification strategy for solving a large-scale static DARP (*16*).

This research extends the work of Fu (*11, 12*) by considering (*a*) a more-realistic objective function and complex constraints for DARP that were not covered in Fu's work and (*b*) multiple depots. Most studies related to DARPs assume the availability of a fleet of *m* vehicles based at a single depot to simplify the complicated real-world problem. The real world may include several depots, especially in wide geographical areas like metropolitan cities in which a dial-a-ride service is provided by two or more private companies. The proposed model has a (*a*) comprehensive objective function that combines service provider's cost and customers' cost and (*b*) complex constraints to explain and reflect real-world operation more reasonably than other models. The proposed model can get more reasonable and flexible solutions by using soft time windows that are allowed to be violated subject to penalties than solutions available by means of hard time windows.

Therefore, this paper is mainly focused on developing such algorithms and solution approaches for static multidepot DARPs with time-varying travel times and soft time windows. The remainder of this paper is organized as follows. The next section describes the problem and mathematical formulation. Then the focus is on the heuristic algorithm for solving the model. Following that results of a computational study are presented. Finally, the last section summarizes the results and provides ideas for further research.

PROBLEM DESCRIPTION AND FORMULATION

Problem Description

As noted earlier, this paper is focused on static DARP with time-varying travel times, multidepots, and heterogeneous vehicles.

This problem considers accommodating only one type of demand that is known in advance because the customers have the ability to make reservations for service on a particular day in advance. It also considers time-varying travel times because link flow speeds are not fixed during the service period and fluctuate. Each customer has a pickup and a drop-off time window. As in real-world situations, the problem considers more than one depot from which vehicles of a fleet can start operating.

Demands

All demands are known in advance because of customers' reservations before the trip day. After a demand is accepted, it cannot be canceled. Each demand has both pickup and drop-off service. Every demand has a demand request time (a desired pickup time and drop-off time), pickup node, drop-off node, and load (regular passengers, wheelchair passengers, or transferrable wheelchair passengers). A customer is picked up from the pickup node and transported to the drop-off node by the same vehicle. There are no customer priorities, and passengers cannot be transferred between vehicles.

Travel Time

Travel times are subject to change by the time of day. Travel times from one location to another are not necessarily the same in both directions. The proposed model assumes link flow speeds, which is based on historical data in the network, within each time interval (10 min). Given link flow speeds, the expected travel time between origin and destination at the starting time can be calculated by using a time-dependent shortest path algorithm. For a one-to-one time-dependent shortest-path algorithm, the one-to-all Dijkstra's algorithm with double buckets that was used to get the static shortest paths by Cherkassky et al. and Zhan was extended and again used (17, 18). Furthermore, to hold the first-in, first out property, the flow speed model used by Sung et al. was adopted for this problem (19).

Routing Plan

The routing plan is developed on the basis of the demands, which are known in advance. The model assumes a maximum route duration *u*, which cannot be exceeded by any vehicle. In addition, the ride time for a customer who is picked up at pickup node *i* cannot exceed his or her maximum ride time r_i. The maximum ride time for customer *i*, r_i, can be calculated by using a linear function of the direct ride time $R_{i,n+i}$ between pickup node *i* and drop-off node $n + i$ and the constant β, as follows (8):

$$r_i(\text{min}) = \beta(\text{min}) + 1.5 \times R_{i,n+i}(\text{min})$$

In the model, $R_{i,n+i}$ is replaced by $R^t_{i,n+i}$, which is the direct ride time between pickup node *i* and drop-off node $n + i$ at time *t* on the basis of the time-varying travel times.

Depots

The model assumes multiple depots. The number of available vehicles at a depot and locations of depots are known. When a vehicle completes its service, it has to return to the depot to which the vehicle belongs. Relocation of vehicles between depots is not allowed. This constraint is consistent with paratransit operations of the Baltimore, Maryland, area. The Maryland Transit Administration and two private transit companies operate paratransit services in this area, and each company has its own depot.

Vehicles

Each vehicle has its own capacity, and the vehicles are not homogeneous. Because most dial-a-ride services are provided to handicapped patients who are transported to and from hospitals and medical facilities, it is important to distinguish the patient types. Two types of handicapped patients are considered in this research: ambulatories who use regular seats and those who use wheelchairs. This constraint is consistent with industry norms.

The model assumes that the capacities of all vehicles are known. The number of vehicles used for services is minimized because the fixed cost of using additional vehicles in the real world is much higher than the routing costs.

Time Windows

Each customer who uses the service has a time window for pick-up and a time window for drop-off. The time window at demand node i is denoted by (a_i, b_i), where ai is the earliest allowable arrival time and b_i is the latest allowable arrival time. In this paper, we consider soft time windows and allow vehicles to arrive at the pick-up and drop-off nodes before or after the time interval that is designated for service. The early and late arrivals at a location are penalized so that the violation of time windows is kept to a minimum. The early arrival penalty is applied when a vehicle arrives before a_i, and the delay penalty is applied when a vehicle arrives after b_i. This is more realistic than hard time windows and is consistent with real-world operations.

Although it is possible that time windows can be negotiated between the customers and the scheduler, in practice, fixed time windows are normally adopted. If any customer is aboard a vehicle, the driver is not allowed to wait for servicing customers at the node.

Problem Formulation

This section provides a mathematical formulation for static multidepot DARPs with time windows as a mixed-integer programming problem.

Notation and Variables

The data sets, constraints, and decision variables used in this model formulation are defined follows.

Data Sets The data sets are designated by the following notation:

$D(k, i, j)$ = demand set, where k = identity number of demand, i = pickup node, j = drop-off node;
I = set of demand identification numbers {set of k in $D(k, i, j)$};
P = set of pick-up nodes {set of i in $D(k, i, j)$} = {1, 2, 3, . . . , n};
B = set of drop-off nodes {set of j in $D(k, i, j)$} = {1 + 1, 2 +1, 3 + 1, . . . , n + 1};
ϕ = total pickup and drop-off node set ($P \cup B$);
$\delta(k, l, i, j)$ = vehicle information set, where k = vehicle number, l = depot number, i = starting node, j = ending node;
V = vehicles set {set of k in $\delta(k, l, i, j)$};
H = set of depots {set of l in $\delta(k, l, i, j)$};
S = set of starting nodes of all vehicles {set of i in $\delta(k, l, i, j)$};
E = set of ending nodes of all vehicles {set of j in $\delta(k, l, i, j)$};
K = total vehicles;
η = union of the set of all demand nodes and the set of all starting nodes ($\phi \cup S$);
N = set of all nodes ($\phi \cup S \cup E$);
N_s = set of all nodes except the starting nodes ($N - S$); and
D_s = set of all demand nodes except the starting nodes ($\phi - S$).

Constants Constants are designated by the following notation:

f_c = fixed cost of a vehicle,
P_e = excess time penalty caused by excess ride time of customer,
P_w = waiting penalty caused by early arrival at each demand node,
P_d = delay penalty caused by late arrival at each demand node,
R_c = traveling cost per unit time,
t = time counter,
α = starting time of the service period,
ω = end time of the service period,
R^t_{ij} = time-dependent shortest travel time from demand node i to j at starting time t,
a_i = earliest allowable arrival time at demand node i,
b_i = latest allowable arrival time at demand node i,
s_i = dwell time needed at demand node i for boarding or alighting,
$w_{\max}$ = maximum acceptable waiting time at each demand node,
$d_{\max}$ = maximum acceptable delay time at each demand node,
u = maximum route duration for all vehicles,
r_i = maximum ride time for passenger with pickup mode i,
(q^a_i, q^w_i) = load of ambulatory passengers, wheelchair passengers, at demand node i (if demand node i is a pickup service, then q_i is a positive value; if demand node i is a drop-off service, then q_i is a negative value), and
(C^a_k, C^w_k) = two-dimensional capacity of vehicle k.

Decision Variables Decision variables are designated by the following notation:

$$x^{kt}_{ij} = \begin{cases} 1 & \text{if vehicle } k \text{ departs from node } i \text{ to node } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

w_i = waiting time at demand node i (desired arrival time at demand node i – actual arrival time at demand node i),
d_i = delayed time at demand node i (actual arrival time at demand node i – desired arrival time at demand node i),
(Q^a_{ik}, Q^w_{ik}) = actual load of vehicle k when departing demand node i (ambulatory seats and wheelchair seats), and

$$y_{ik} = \begin{cases} 1 & \text{if vehicle } k \text{ arrives with passengers on board at node } i \\ 0 & \text{otherwise} \end{cases}$$

The proposed mathematical formulation is as follows:

$$\min\ f_c \times \sum_{k\in K}\sum_{i\in S}\sum_{j\in N_s}\sum_{t=\alpha}^{\omega} x^{kt}_{ij} + R_c \times \sum_{i\in\tau}\sum_{j\in N_s}\sum_{k\in K}\sum_{t=\alpha}^{\omega}\left(R^t_{ij} \times x^{kt}_{ij}\right) + P_w \times \sum_{i\in\phi} w_i + P_e \times \sum_{i\in P}\left(\sum_{k\in K}\sum_{l\in N_s}\sum_{t=\alpha}^{\omega}(t - s_{i+1})x^{kt}_{i+1,l} - \sum_{k\in K}\sum_{j\in\phi}\sum_{t=\alpha}^{\omega} t x^{kt}_{ij} - \sum_{t=\alpha}^{\omega}\left(R^t_{i,i+1} \times \sum_{k\in K}\sum_{j\in\phi} x^{kt}_{ij}\right)\right) + P_d \times \sum_{i\in\phi} d_i \qquad (1)$$

subject to

$$\sum_{i\in S}\sum_{j\in P}\sum_{t=\alpha}^{\omega} x^{kt}_{ij} \le 1 \qquad k \in V \qquad (2)$$

$$\sum_{j\in E}\sum_{i\in B}\sum_{t=\alpha}^{\omega} x^{kt}_{ij} \le 1 \qquad k \in V \qquad (3)$$

$$\sum_{k\in K}\sum_{i\in B}\sum_{t=\alpha}^{\omega}\left(x^{kt}_{ij}\left(t + R^t_{ij}\right)\right) \le \omega \qquad j \in E \qquad (4)$$

$$Q^a_{ik} \le \sum_{j\in\phi}\sum_{t=\alpha}^{\omega} x^{kt}_{ij} C^a_k \qquad k \in K, i \in \phi \qquad (5a)$$

$$Q^w_{ik} \le \sum_{j\in\phi}\sum_{t=\alpha}^{\omega} x^{kt}_{ij} C^w_k \qquad k \in K, i \in \phi \qquad (5b)$$

$$Q_{ik}^{a} + q_{j}^{a} - Q_{jk}^{a} - M \times \left(1 - \sum_{t=\alpha}^{\omega} x_{ij}^{kt}\right) \leq 0 \qquad k \in K, i \in \eta, j \in N_s, i \neq j \tag{6a}$$

$$Q_{ik}^{a} + q_{j}^{a} - Q_{jk}^{a} + M \times \left(1 - \sum_{t=\alpha}^{\omega} x_{ij}^{kt}\right) \geq 0 \qquad k \in K, i \in \eta, j \in N_s, i \neq j \tag{6b}$$

$$Q_{ik}^{w} + q_{j}^{w} - Q_{jk}^{w} - M \times \left(1 - \sum_{t=\alpha}^{\omega} x_{ij}^{kt}\right) \leq 0 \qquad k \in K, i \in \eta, j \in N_s, i \neq j \tag{7a}$$

$$Q_{ik}^{w} + q_{j}^{w} - Q_{jk}^{w} + M \times \left(1 - \sum_{t=\alpha}^{\omega} x_{ij}^{kt}\right) \geq 0 \qquad k \in K, i \in \eta, j \in N_s, i \neq j \tag{7b}$$

$$\sum_{k\in K}\sum_{j\in\phi}\sum_{t=\alpha}^{\omega} x_{ij}^{kt} = 1 \qquad i \in P \tag{8}$$

$$\sum_{j\in\phi}\sum_{t=\alpha}^{\omega} x_{ij}^{kt} - \sum_{l\in\phi}\sum_{t=\alpha}^{\omega} x_{l,i+1}^{kt} = 0 \qquad k \in K, i \in P \tag{9}$$

$$\sum_{l\in\phi}\sum_{t=\alpha}^{\omega}\left(x_{l,i+1}^{kt}\left(t + R_{l,i+1}^{t}\right)\right) - \sum_{j\in\phi}\sum_{t=\alpha}^{\omega}\left(x_{ij}^{kt}\left(t + R_{ij}^{t}\right)\right) \geq 0 \qquad k \in K, i \in P \tag{10}$$

$$\sum_{\substack{l\in N_s\\ j\neq l}}\sum_{t=\alpha}^{\omega} x_{jl}^{kt} - \sum_{\substack{i\in\eta\\ i\neq j}}\sum_{t=\alpha}^{\omega} x_{ij}^{kt} = 0 \qquad k \in K, j \in D_s \tag{11}$$

$$\sum_{\substack{l\in N_s\\ j\neq l}}\sum_{t=\alpha}^{\omega} tx_{jl}^{kt} - s_j - w_j - \sum_{\substack{i\in\eta\\ i\neq j}}\sum_{t=\alpha}^{\omega}\left(x_{ij}^{kt}\left(t + R_{ij}^{t}\right)\right) + M\left(1 - \sum_{\substack{i\in\eta\\ i\neq j}}\sum_{t=\alpha}^{\omega} x_{ij}^{kt}\right) \geq 0 \qquad k \in K, j \in \phi \tag{12a}$$

$$\sum_{\substack{l\in N_s\\ j\neq l}}\sum_{t=\alpha}^{\omega} tx_{jl}^{kt} - s_j - w_j - \sum_{\substack{i\in\eta\\ i\neq j}}\sum_{t=\alpha}^{\omega}\left(x_{ij}^{kt}\left(t + R_{ij}^{t}\right)\right) - M\left(1 - \sum_{\substack{i\in\eta\\ i\neq j}}\sum_{t=\alpha}^{\omega} x_{ij}^{kt}\right) \leq 0 \qquad k \in K, j \in \phi \tag{12b}$$

$$\sum_{l\in B}\sum_{m\in E}\sum_{t=\alpha}^{\omega}\left(x_{lm}^{kt}\left(t + R_{lm}^{t}\right)\right) - \sum_{i\in S}\sum_{j\in P}\sum_{t=\alpha}^{\omega}\left(tx_{ij}^{kt}\right) \leq u \qquad k \in V \tag{13}$$

$$\sum_{k\in V}\sum_{l\in N_s}\sum_{t=\alpha}^{\omega} tx_{i+1,l}^{kt} - \sum_{k\in V}\sum_{j\in\phi}\sum_{t=\alpha}^{\omega} tx_{ij}^{kt} \leq r_i \qquad i \in P \tag{14}$$

$$w_j = \max\left(0, a_j - \sum_{k\in K}\sum_{i\in\eta}\sum_{t=\alpha}^{\omega}\left(x_{ij}^{kt}\left(t + R_{ij}^{t}\right)\right)\right) \qquad j \in \phi \tag{15a}$$

$$0 \leq w_j \leq w_{\max} \qquad j \in \phi \tag{15b}$$

$$d_j = \max\left(0, \sum_{k\in K}\sum_{i\in\eta}\sum_{t=\alpha}^{\omega}\left(x_{ij}^{kt}\left(t + R_{ij}^{t}\right)\right) - b_j\right) \qquad j \in \phi \tag{16a}$$

$$0 \leq d_j \leq d_{\max} \qquad j \in \phi \tag{16b}$$

$$M\left(1 - y_{ik}\right) \geq w_i \qquad k \in K, i \in \phi \tag{17}$$

$$My_{ik} \geq Q_{ik}^{a} - q_{i}^{a} + Q_{ik}^{w} - q_{i}^{w} \qquad k \in K, i \in P \tag{18a}$$

$$M\left(y_{ik} - 1\right) \leq Q_{ik}^{a} - q_{i}^{a} + Q_{ik}^{w} - q_{i}^{w} - 1 \qquad k \in K, i \in P \tag{18b}$$

$$My_{ik} \geq Q_{ik}^{a} + Q_{ik}^{w} \qquad k \in K, i \in B \tag{18c}$$

$$M\left(y_{ik} - 1\right) \leq Q_{ik}^{a} + Q_{ik}^{w} - 1 \qquad k \in K, i \in B \tag{18d}$$

where M is a large positive number.

Objective Function

The objective of this problem is to minimize the total cost, composed of the service provider's cost and the customers' inconvenience cost. First, an attempt is made to minimize the service provider's cost, including the fixed costs for the used vehicles, the routing costs, and vehicle waiting cost. Second, the user inconvenience cost, including customers' excess ride time cost and delayed service cost, are minimized.

Constraints

The constraints in this model can be divided into five groups: depot, capacity, precedence and coupling, routing, and time window.

Depot Constraints (Equations 2 Through 4) The depot constraints require that vehicles start and end in the depot to which they belong. Every vehicle has to return to the depot before the end of the service period.

Capacity Constraints (Equations 5 Through 7) Each vehicle has its own capacity (C_k^a, C_k^w) for ambulatory passengers and wheelchair passengers. These capacities cannot be exceeded at any time.

Precedence and Coupling Constraints (Equations 8 Through 10) The precedence and coupling constraints represent the requirement that each customer must first be picked up at node i and then dropped off at node $i + 1$ by the same vehicle k. Each demand node is visited exactly once during a day.

Routing Constraints (Equations 11 Through 14) When a vehicle arrives at a node that is not a depot, it has to travel to either another demand node or a depot (route continuity). If demand node j is visited after visiting demand node i, then the arrival time at demand node j must be equal to the sum of the departure time at demand node i and the travel time R_{ij}^t from demand node i to demand node j at departure time t at demand node i. The route duration for each vehicle cannot exceed the maximum route duration. The ride time for a customer who is picked up at node i cannot exceed the maximum ride time r_i.

Time Window Constraints (Equations 15 and 16) The waiting time at a demand node j is the gap between the earliest arrival time and the actual arrival time at the demand node j. The waiting time cannot exceed the maximum acceptable waiting time $w_{\max}$. The delay time at a demand node j is the gap between the latest arrival time and the actual arrival time at the demand node j. The delay time cannot exceed the maximum acceptable delay time $d_{\max}$.

No-Waiting-on-Board Constraints (Equations 17 and 18) If any customer is aboard the vehicle, the driver is not allowed to wait for servicing customers at the node.

HEURISTIC ALGORITHMS FOR STATIC MULTIDEPOT DARP

In this paper, the approach for solving the model can be divided into two phases: a construction phase and an improvement phase. In the first phase, feasible routes are constructed, and in the second phase, the routes are improved. Three heuristic algorithms—one based on sequential insertion (HSI), one based on parallel insertion (HPI), and one based on clustering first–routing second (HCR)—are proposed. These differ in the the way that they construct feasible routes. The framework of this basic heuristic is depicted in Figure 1.

Construction

Before the construction phase, all demands are grouped into the closest depot and sorted by earliest allowable time window.

In the construction phase, a set of feasible routes is built for each depot, starting from the information that defines the DARP by the three heuristic methods: HSI, HPI, and HCR.

For HSI

The procedure is similar to the one proposed in Jaw et al. (*8*). The algorithm starts by sorting customers in increasing order of their pickup times and inserts one customer at a time into one vehicle's schedule. The algorithm used in this paper proceeds as follows:

1. Sort demands by their earliest time window and create List L.
2. Construct Conflict Table C. If customers *i* and *j* cannot be serviced in one trip in the worst case, they are marked as conflicting.
3. Cluster demands according to List L and the Conflict Table C. Unvisited requests are clustered into different groups.
4. Construct a feasible route by sequentially extracting as many customers as possible from one route.
5. After extracting one route, regroup the remaining requests in the route into a new route.
6. Construct feasible routes until no customers are left.

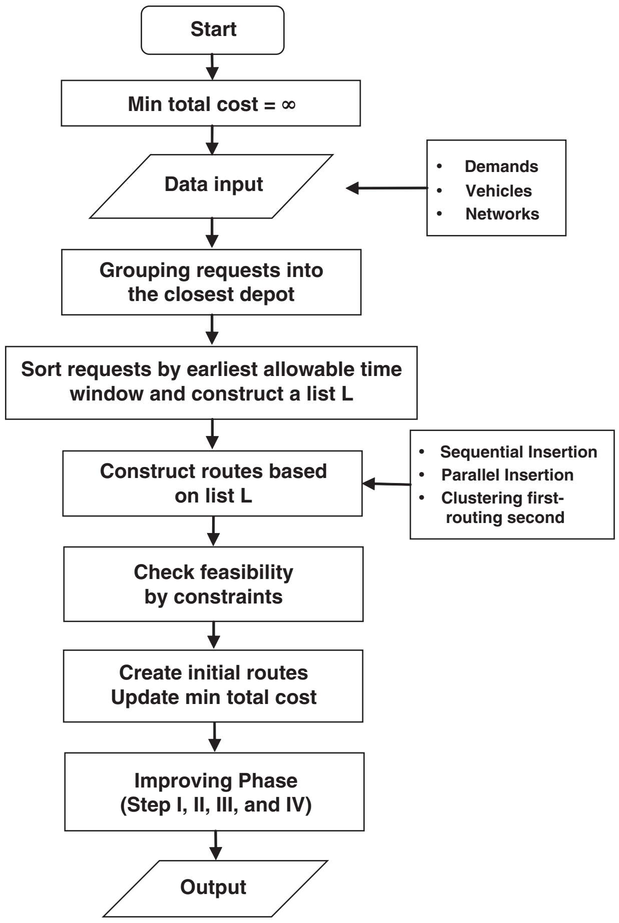
Start
Min total cost = ∞
Data input
• Demands
• Vehicles
• Networks
Grouping requests into the closest depot
Sort requests by earliest allowable time window and construct a list L
Construct routes based on list L
• Sequential Insertion
• Parallel Insertion
• Clustering first-routing second
Check feasibility by constraints
Create initial routes Update min total cost
Improving Phase (Step I, II, III, and IV)
Output

FIGURE 1 Framework of heuristic.

For HPI

The procedure is similar to the one proposed by Toth and Vigo (*10*). First, a small set of empty routes is initialized, and then iteratively unscheduled demands are inserted into the existing route that has the cheapest insertion cost for those demands. The algorithm proceeds as follows:

1. Sort demands by their earliest time window and create List L.
2. Construct Conflict Table C. If customers *i* and *j* cannot be serviced in one trip in the worst case, they are marked as conflicting.
3. Initialize a set of empty routes, M.
4. Then each customer is processed in the list in sequence as follows and assigned to a vehicle until the list of customers is exhausted. Process each customer in the list in sequence as follows and assign him or her to a vehicle until List L is exhausted. For each customer i ($i = 1, 2, \dots, N$) and each route j ($j = 1, 2, \dots, M$),
   – Find all feasible insertion sequences in which customer *i* can be inserted into route *j*. If it is infeasible to assign customer *i* to route *j*, examine the next route $j + 1$, and restart this step; otherwise, find the insertion of customer *i* into the route *j* that results in minimum additional cost. Call this additional cost, *Cj*.
   – If it is infeasible to insert customer *i* into any route *j*, then make a new route, $M + 1$, and insert customer *i* into that route; otherwise, assign customer *i* to route j^*, which has a minimum additional cost for all j ($j = 1, 2, \dots, M$).

For HCR

In this approach, first, clusters are made and a vehicle is assigned to each cluster. Then, customers in the group are assigned to each cluster. The method for making clusters for customers is as follows:

1. Calculate an urgency index value for each customer *i*:

$$U_i = \alpha \times \frac{d1_i}{\max d1} - \beta \times \frac{\mathrm{ET}_i}{\max \mathrm{ET}} + \gamma \times \frac{d2_i}{\max d2}$$

where

$d1_i$ = Euclidean distance between pickup and drop-off node of customer *i*,
$d2_i$ = Euclidean distance between depot and pickup node of customer *i*, and
ET_i = earliest allowable arrival time of customer *i*.

2. Sort customers by index value and make a list.
3. Choose a seed (the starting customer) from the list for each cluster. The customers are chosen in order from the list so as to create a set of clusters equal to the minimum number of routes that have been set.
4. Assign vehicles to seeds. Vehicles are randomly chosen, but they should accept the type (ambulatory or wheelchair) of the assigned customer.

After the clusters are made, the method for adding any unassigned customers to clusters is as follows:

1. Add other customers in the group to clusters one by one according to geographical closeness until every customer belongs to one of the clusters.
2. For unassigned customer *i*,
   – Calculate the Euclidean distance between unassigned customer *i* and last added customer *j* in each cluster.

$$d_{ij} = \text{distance}\left(\text{pickup node of } i \text{ and pickup node of } j\right) + \text{distance}\left(\text{drop-off node of } i \text{ and drop-off node of } j\right)$$

   – Compare the distances of clusters.
   – Add customer *i* to the cluster that has the shortest distance.

Improvement

After a feasible solution is obtained in the construction phase, it is improved through four steps in the improvement phase.

Step I. Acceptable Waiting and Delay Time

Although violation of the time window is allowed in this model, service quality can be adjusted by the maximum allowable waiting and delay (MaxWD) times in this phase. In the initial solution, after the construction phase, bad routes in which some customers have unreasonable waiting or delay time may exist. In this phase, all routes are improved to satisfy MaxWD times. The method is described as follows:

1. After checking waiting and delay times at each node, choose nodes that have larger values than MaxWD times, which are set in advance.
2. Remove those nodes from the original route, and insert them into another route that does not conflict with them.
3. If no route can accept them, make a new route.

Step II. Remove One, Insert One

For the local improvement procedure to explore the local region, a trip insertion operator is applied at this phase. Trip insertion removes a given trip from a route and then inserts it into the best position of another route. For example, through this method, Customer 3 (Demand Nodes 5 and 6) is inserted in Route 2 (as shown in Figure 2), which produces a smaller additional insertion cost.

The method is described in detail as follows:

1. At a route (original route), choose one pair (pickup and drop-off nodes); search all possible routes (target routes) that can accept it; and insert it into each target route. Find the best position in each target route.
2. Calculate saving cost (SC) for the original and each target route. From all target routes, choose the one that has the best SC (it should be positive), and then update the original and target routes and minimum total cost.

$$\text{SC}(i, j) = f(i) + f(j) - f(i') - f(j')$$

where

$f(i)$ = cost of route *i*,
i = original route before moving,
j = target route before moving,
i' = original route after moving, and
j' = target route after moving.

3. If insertion of one pair into other routes fails or best SC is negative, insert the pair back into its original route.
4. Repeat Steps 1 through 3 of this sequence for other pairs in the original route.
5. Repeat Steps 1 through 4 of this sequence for other routes.

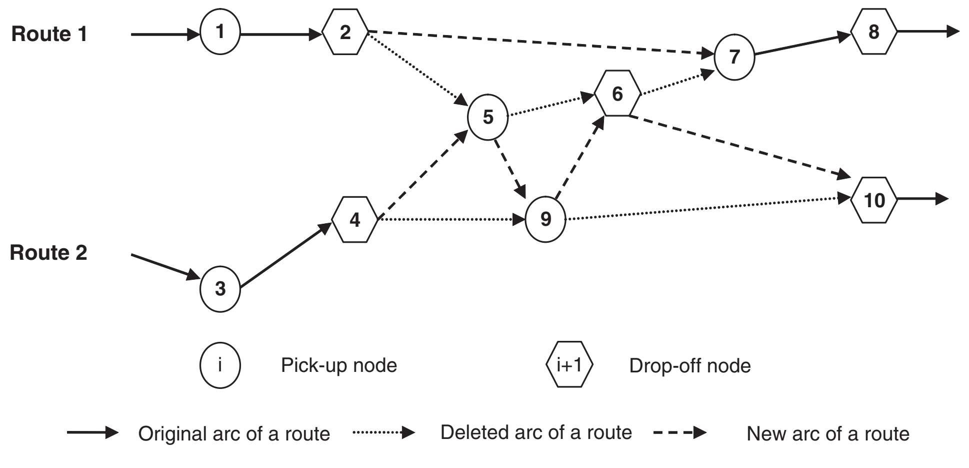

FIGURE 2 Illustration of remove-one, insert-one method.

Improvement Step II is repeated up to two times because it has been observed that, in most cases, after two repetitions, the objective functions converge.

Step III. Combining Vehicles

After Improvement Step II, some routes may have a few demands. These routes can be combined into other routes that can accept customers without violating constraints. A vehicle may be combined into another vehicle that starts from a different depot.

The procedure of combining vehicles proceeds as follows:

1. Choose a route that has fewer demands than min_demands to guarantee that each route has at least some demands.
2. Find other routes into which the chosen route can be combined, and calculate SC for each case.
3. Combine the chosen route into the route that has maximum SC.

Step IV. Adjusting Vehicle Starting Times

The starting time of a route is determined simply to set the departure time from the depot as the earliest arrival time at the first pickup node minus the travel time between the depot and that node. Still, there is a possibility to reduce the waiting time and the total route duration by adjusting vehicle starting time at the depot. The method proposed by Xing et al. is used for this step (*16*).

At this step, the waiting time is checked at a demand node in each route. If any waiting time exists at a demand node in a route, the starting time of the vehicle serving that route is adjusted by using marginal time as follows: for each route i ($i = 1, 2, \dots, M$),

1. Check the waiting time at each demand node in route i.
2. If any waiting time exists at a demand node in route i, then calculate marginal time at each node in route i. The marginal time at a node is defined as the maximum delay in arrival at the current node that does not cause violation of the time windows at the following nodes. Otherwise, go to next route, $i + 1$, and repeat Step 1 of this sequence.
3. Adjust the vehicle starting time to the earliest arrival time at the first pickup node minus the travel time between the depot and that node plus the marginal time at the first node in route i.

COMPUTATIONAL STUDY

In this section, first, the model is validated through the solution of a set of small test problems by an exact method that uses the commercial package CPLEX. In addition, the results of the exact method are compared with the results of the heuristics algorithm that is developed for the model in this paper. Second, the results of the three algorithms based on HCI, HSI, and HCR are compared in relation to problem instances that have 30, 50, and 100 customers, respectively. Finally, the performance of the three heuristic algorithms is analyzed in this section.

The heuristic algorithms were coded in C++. All the following computations were completed on a machine with 2.0-GHz Intel Core 2 Duo CPU and 3-GB memory in the Windows XP environment.

Test Problems for Validating Model and Heuristic Algorithms

Characteristics of Problem Instances

The exact method can solve problems with a few customers that have to be serviced with a few vehicles. The service area was assumed to be 20×20 mi with two depots. The location of Depot 1 was (7, 10) and of Depot 2 was (13, 10). The demands were generated at random over the service area. There were three, four, and five customers with 10 and 15 time intervals, respectively. Interval length was 6 min. For the 10 time-interval case, the period was from 9 to 10 a.m. For the 15 time-interval case, the period was from 9 to 10:30 a.m.

Parameter Settings

The duration of the time window was 12 min; maximum route duration for 10 time intervals was 60 min and for 15 time intervals was 90 min; MaxWD times were 30 min; fixed cost for each used vehicle was \$10,000; travel cost was \$1/min; penalty cost for waiting time was \$0.50/min; penalty cost for delay time was \$0.50/min; and penalty cost for customers' excess ride time was \$0.50/min.

Computational Results 1

For small problems, three heuristic algorithms solved the test problems faster than the exact method with almost the same objective function values. Table 1 shows the computational results for the small test problems.

As the number of customers exceeded three with a service period of 10 time intervals, the calculation time of the exact method increased exponentially and became unreasonable. The largest DARP problem size that could be solved in a reasonable time by the exact method was five customers with a service period of 10 time intervals. The gaps of the objective function values between the exact method and three heuristic algorithms were less than 0.003%. For most of the cases, three heuristic algorithms solved the problems within less than 0.2 s, although the exact method could not solve the problem that had five customers and 15 time intervals. For example, in the case of five customers and 10 time intervals, HPI solved the problem within 0.12 s, HSI within 0.16 s, and HCR within 0.17 s, whereas the exact method solved the problem within about 14 min.

Of the three heuristic algorithms, the HPI had the best performance in relation to calculation times and objective function values. For objective function values, HPI, HSI, and HCR had almost the same objective function values within less than a 0.004% gap for all cases, and HPI had a little better objective function value than HSI and HCR. For calculation times, subtle differences occurred between three heuristic algorithms, and HPI solved the problems a little faster than HSI or HCR.

Test Problems for Analyzing Performance of Heuristic Algorithm

To test larger size problems that are similar to real service, several test problems were generated at random. The three heuristic algorithms were applied to these problems, and their performance was compared.

TABLE 1 Computational Results of Small Test Problems

		Calculation Time (s)					Objective Function				
Number of Customers	Number of Time Interval	CPLEX (*a*)	HCR (*b*)	HSI (*c*)	HPI (*d*)	Ratio (*a/d*)	CPLEX (*a*)	HCR (*b*)	HSI (*c*)	HPI (*d*)	Gap ((*d* – *a*)/ *a* ∗ 100) (%)
3	10	8.08	0.16	0.15	0.14	56.4	30,015.3	30,015.7	30,017.0	30,015.7	0.001
3	15	67.87	0.16	0.14	0.14	500.7	30,019.2	30,020.0	30,019.7	30,019.7	0.002
4	10	56.30	0.17	0.17	0.15	372.8	40,023.3	40,023.7	40,023.7	40,023.7	0.001
4	15	1,410.43	0.12	0.12	0.11	12,991.4	36,689.0	36,690.7	36,690.0	36,690.0	0.003
5	10	867.18	0.17	0.16	0.12	7,115.8	50,028.3	50,028.8	50,028.8	50,028.8	0.001
5	15	—	0.15	0.13	0.12	—	—	50,029.0	50,028.8	50,028.8	—

Characteristics of Problem Instances

The service area was assumed to be 20 × 20 mi with two depots. The location of Depot 1 was and of Depot 2 was (*7, 10, 13*). The demands were generated at random over the service area. The problem sizes were 30, 50, and 100 customers, respectively. Time periods were from 6 a.m. to 6 p.m. There were 72 time intervals, and each was 10 min.

Parameter Settings

The width of the time window was 30 min; maximum route duration was 720 minutes; MaxWD times were 30, 20, and 10 min; fixed cost for each used vehicle was $500; travel cost was $1/min; penalty cost for waiting time was $0.50/min; penalty cost for delay time was $0.50/min; and penalty cost for customers' excess ride time was $0.50/min. The service times at demand node were 2 min for a regular passenger and 6 min for a passenger with a wheelchair.

Link Flow Speeds

In the test problems, three classes of roads were assumed in the network: first, highways with a speed limit of 60 mph; second, major roads with a speed limit of 40 mph; and third, minor roads with a speed limit of 30 mph. Each link belongs to one of these classes. It was also assumed that link flow speeds of highways and major roads (though not the minor roads) varied according to time interval.

Computational Results 2

As the sizes of the problems increased, the calculation times for solving them increased exponentially. In most cases, the problems were solved within 2 min by means of the heuristic algorithms. The calculation times, objective functions, and problem sizes are described in Table 2.

As mentioned earlier, MaxWD are considered in this research, and the behavior of the proposed three heuristics was evaluated under three scenarios in which MaxWD was 30, 20, and 10 min, respectively (Table 2). HSI performed better than HCR or HPI in relation to calculation time in most cases. For example, in the case of 100 customers, HSI solved the problem 42.4% faster than HCR for a MaxWD of 30 min, 27.5% faster for a MaxWD of 20 min, and 34.8% faster for a MaxWD of 10 min. Even in the worst cases, the difference between the calculation time of HSI and HCR was less than −2.0%. In addition, the difference between the calculation times of HSI and HCR and the difference between the calculation times of HSI and HPI decreased as MaxWD decreased from 30 to 10 min for all cases of 30, 50 and 100 customers.

Furthermore, when the objective function values were considered, HSI was better than HPI or HCR in most cases. For example, in the case of 100 customers, the solution of HSI was 26.7% better than that of HCR for a MaxWD of 30 min, 9.8% better for a MaxWD of 20 min, and 9.0% better for a MaxWD of 10 min. Even in the worst cases, the difference between the objective function values of HSI and HCR was less than −5%.

TABLE 2 Calculation Times, Objective Functions, and the Problem Sizes

			Calculation Time (ms)					Objective Function				
Number of Customers	MaxWD	Vehicles at Each Depot	HSI (*a*)	HPI (*b*)	HCR (*c*)	Savings ((*b* – *a*)/ *b* ∗ 100) (%)	Savings ((*c* – *a*)/ *c* ∗ 100) (%)	HSI (*a*)	HPI (*b*)	HCR (*c*)	Savings ((*b* – *a*)/ *b* ∗ 100) (%)	Savings ((*c* – *a*)/ *c* ∗ 100) (%)
30	30	15	2,782	4,184	3,428	33.5	18.87	3,651	6,011	5,021	39.3	27.3
30	20	15	3,138	3,666	3,173	14.4	1.1	5,149	5,542	5,043	7.1	–2.1
30	10	15	2,972	3,446	2,915	13.8	–2.0	5,595	6,074	6,043	7.9	7.4
50	30	30	7,063	8,460	8,673	16.5	18.6	5,686	6,301	8,088	9.8	29.7
50	20	30	7,904	8,771	7,927	9.9	0.3	8,016	6,934	7,650	–15.6	–4.8
50	10	30	8,036	7,590	7,993	–5.9	–0.5	8,506	7,958	8,661	–6.9	1.8
100	30	99	51,647	68,469	89,682	24.6	42.4	12,979	15,825	17,704	18.0	26.7
100	20	99	52,661	58,768	72,605	10.4	27.5	14,947	16,359	16,578	8.6	9.8
100	10	99	51,204	55,670	78,515	8.0	34.8	16,041	15,948	17,625	–0.6	9.0

TABLE 3 Performance of Three Heuristic Algorithms: 30-Customer Case

	MaxWD = 30			MaxWD = 20			MaxWD = 10		
Variable	HSI	HPI	HCR	HSI	HPI	HCR	HSI	HPI	HCR
Used vehicle	5	10	8	8	9	8	9	10	10
Serviced customers	30	30	30	30	30	30	30	30	30
Avg. serviced customers per vehicle	6.0	3.0	3.75	3.75	3.33	3.75	3.33	3.00	3.00
Number of serviced ambulatory	48	48	48	48	48	48	48	48	48
Number of serviced wheelchair	10	10	10	10	10	10	10	10	10
Avg. travel times per vehicle (min)	207.40	93.10	118.13	130.75	108.33	116.00	115.00	101.00	98.40
Avg. travel times per customer (min)	34.57	31.03	31.50	34.87	32.50	30.93	34.50	33.67	32.80
Avg. waiting times per vehicle (min)	15.20	1.30	7.13	3.50	2.56	5.50	2.11	1.20	0.00
Avg. delay times per customer (min)	3.13	2.50	0.13	1.03	2.97	3.30	1.63	0.83	1.20
Avg. excess ride times per customer (min)	1.97	2.40	3.03	2.93	0.73	2.90	1.77	3.00	2.73

In addition, the performance of the dial-a-ride service—on the basis of the results of routing and scheduling—as determined by the three heuristic algorithms is shown in Tables 3, 4, and 5.

For most cases, the number of vehicles needed in the HPI or HCR solution was larger than that in the HSI solution. For 30-customer cases and 30 min of MaxWD, 10 vehicles were used in the HPI solution, whereas five vehicles were used in HSI solution. Further, average travel time per vehicle and average travel time per customer in the HSI solution were larger than those in the HPI or HCR solution for all 30-, 50-, and 100-customer cases.

In this research, the initial solution was improved through a series of four steps. As Figure 3 shows, most improvements in the objective

TABLE 4 Performance of Three Heuristic Algorithms: 50-Customer Case

	MaxWD = 30			MaxWD = 20			MaxWD = 10		
Variable	HSI	HPI	HCR	HSI	HPI	HCR	HSI	HPI	HCR
Used vehicle	7	9	13	12	10	12	13	12	14
Serviced customers	50	50	50	50	50	50	50	50	50
Avg. serviced customers per vehicle	7.14	5.56	3.85	4.17	5.00	4.17	3.85	4.17	3.57
Number of serviced ambulatory	73	73	73	73	73	73	73	73	73
Number of serviced wheelchair	26	26	26	26	26	26	26	26	26
Avg. travel times per vehicle (min)	278.72	175.78	111.46	154.42	178.00	128.00	147.69	153.42	113.43
Avg. travel times per customer (min)	39.02	31.64	28.98	37.06	35.60	30.72	38.40	36.82	31.76
Avg. waiting times per vehicle (min)	15.71	7.67	5.08	2.83	2.60	0.92	0.15	2.17	0.00
Avg. delay times per customer (min)	3.44	3.68	0.86	3.16	2.38	1.76	1.04	1.30	0.82
Avg. excess ride times per customer (min)	3.76	3.70	3.38	2.68	3.26	2.60	2.36	2.88	2.12

TABLE 5 Performance of Three Heuristic Algorithms: 100-Customer Case

	MaxWD = 30			MaxWD = 20			MaxWD = 10		
Variable	HSI	HPI	HCR	HSI	HPI	HCR	HSI	HPI	HCR
Used vehicle	19	25	29	23	26	27	25	25	29
Serviced customers	100	100	100	100	100	100	100	100	100
Avg. serviced customers per vehicle	5.26	4.00	3.45	4.35	3.85	3.70	4.00	4.00	3.45
Number of serviced ambulatory	157	157	157	157	157	157	157	157	157
Number of serviced wheelchair	50	50	50	50	50	50	50	50	50
Avg. travel times per vehicle (min)	170.63	122.04	102.31	141.83	118.92	104.33	135.52	130.52	102.72
Avg. travel times per customer (min)	32.42	30.51	29.67	32.62	30.92	28.17	33.88	32.63	29.79
Avg. waiting times per vehicle (min)	4.32	2.12	1.69	1.04	2.62	2.59	1.84	0.96	0.34
Avg. delay times per customer (min)	1.53	1.77	1.09	1.09	1.50	1.40	0.58	0.83	0.73
Avg. excess ride times per customer (min)	2.40	3.19	5.61	2.38	3.16	3.13	2.10	2.64	2.01

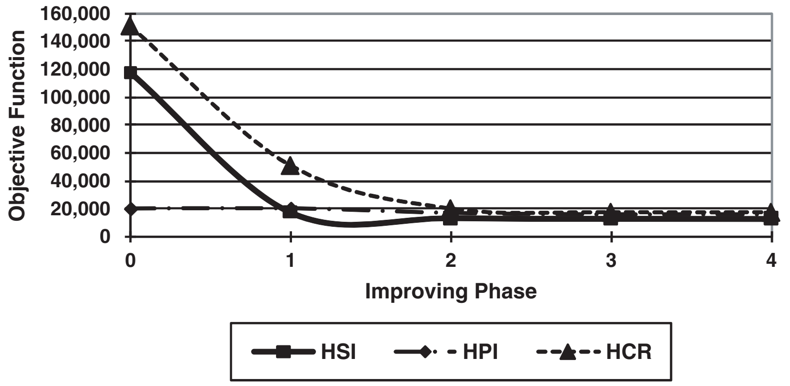

FIGURE 3 Convergence of objective function by improvement phase for 100-customer case and MaxWD of 30 min.

function values for these problems were made through Improvement Steps I and II, and the objective functions converged, as expected.

CONCLUSIONS AND FURTHER STUDY

In this paper, a static DARP with time-varying travel times, soft time windows, multiple depots, and heterogeneous vehicles was formulated as a mixed-integer programming problem. The objective of the formulation was to minimize the total cost, which consisted of the service provider's cost and the customers' inconvenience cost.

A heuristic methodology based on an insertion algorithm and a cluster-first, routing-second method was proposed to solve this problem within a reasonable time for implementation in a real-world situation. The computational results showed that, as the problem sizes increased, the heuristic algorithms were superior to the exact algorithm in calculation time. Furthermore, the heuristic algorithm based on sequential insertion was more efficient in calculation times and objective functions than the two other heuristic algorithms that were based on parallel insertion and clustering first, routing second.

The authors are applying this model to solve a real-world problem that was provided by the Maryland Transit Administration in Baltimore. To solve this problem (which includes 2,400 customers per day) efficiently, faster heuristics are needed. Investigation of a good clustering strategy like the decomposition method is a good avenue for future research. In addition, metaheuristics such as the tabu search, genetic algorithms, and the harmony search can be applied to this problem to improve the solutions.

REFERENCES

1. Baugh, J., G. Kakivaya, and J. R. Stone. Intractability of the Dial-a-Ride Problem and a Multiobjective Solution Using Simulated Annealing. *Engineering Optimization,* Vol. 30, No. 2, 1998, pp. 91–123.
2. Savelsbergh, M. W. P., and M. Solomon. The General Pickup and Delivery Problem. *Transportation Science,* Vol. 29, No. 1, 1995, pp. 17–29.
3. Mitrovic-Minic, S. *Pickup and Delivery Problem with Time Window: A Survey.* SFU CMPT TR 1998-12. Simon Fraser University, Burnaby, British Columbia, Canada, 1998.
4. Mitrovic-Minic, S. *The Dynamic Pickup and Delivery Problem with Time Window.* PhD dissertation. Simon Fraser University, Burnaby, British Columbia, Canada, 2001.
5. Desaulniers, G., J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. VRP with Pickup and Delivery. In *The Vehicle Routing Problem* (P. Toth and D. Vigo, eds.), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Pa., 2002, pp. 225–242.
6. Cordeau, J.-F., and G. Laporte. The Dial-a-Ride Problem: Variants, Modeling Issues and Algorithms. *4OR: A Quarterly Journal of Operations Research,* Vol. 1, No. 2, 2003, pp. 89–101.
7. Cordeau, J.-F., and G. Laporte. The Dial-A-Ride Problem: Models and Algorithms. *Annals of Operations Research,* Vol. 153, No. 1, 2007, pp. 29–46.
8. Jaw, J.-J., A. R. Odoni, H. N. Psaraftis, and N. H. M. Wilson. A Heuristic Algorithm for the Multi-Vehicle Advance Request Dial-a-Ride Problem with Time Windows. *Transportation Research Part B,* Vol. 20, No. 3, 1986, pp. 243–257.
9. Ioachim, I., J. Desrosiers, Y. Dumas, and M. M. Solomon. A Request Clustering Algorithm for Door-to-Door Handicapped Transportation. *Transportation Science,* Vol. 29, No. 1, 1995, pp. 63–78.
10. Toth, P., and D. Vigo. Heuristic Algorithms for the Handicapped Persons Transportation Problem. *Transportation Science,* Vol. 31, No. 1, 1997, pp. 60–71.
11. Fu, L. Improving Paratransit Scheduling by Accounting for Dynamic and Stochastic Variations in Travel Time. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1666,* TRB, National Research Council, Washington, D.C., 1999, pp. 74–81.
12. Fu, L. Scheduling Dial-a-Ride Paratransit Under Time-Varying, Stochastic Congestion. *Transportation Research Part B,* Vol. 36, No. 6, 2002, pp. 485–506.
13. Cordeau, J.-F., and G. Laporte. A Tabu Search Heuristic for the Static Multi-Vehicle Dial-a-Ride Problem. *Transportation Research Part B,* Vol. 37, No. 6, 2003, pp. 579–594.
14. Diana, M., and M. M. Dessouky. A New Regret Insertion Heuristic for Solving Large-Scale Dial-a-Ride Problems with Time Windows. *Transportation Research Part B,* Vol. 38, No. 6, 2004, pp. 539–557.
15. Luo, Y., and P. Schonfeld. A Rejected-Reinsertion Heuristic for the Static Dial-A-Ride Problem. *Transportation Research Part B,* Vol. 41, No. 7, 2007, pp. 736–755.
16. Xiang, Z., C. Chu, and H. Chen. A Fast Heuristic for Solving a Large-Scale Static Dial-a-Ride Problem Under Complex Constraints. *European Journal of Operational Research,* Vol. 174, No. 2, 2006, pp. 1117–1139.
17. Cherkassky, B. B., A. V. Goldberg, and T. Radzik. *Shortest Paths Algorithms: Theory and Experimental Evaluation.* Report 93-1480. Computer Science Department, Stanford University, Calif., 1993.
18. Zhan, F. B. Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures. *Journal of Geographic Information and Decision Analysis,* Vol. 1, No, 1, 1997, pp. 69–82.
19. Sung, K., M. G. H. Bell, M. Seong, and S. Park. Shortest Paths in a Network with Time Dependent-Flow Speeds. *European Journal of Operational Research,* Vol. 121, No. 1, 2000, pp. 32–39.

The Paratransit Committee peer-reviewed this paper.